TECNOLOGIA SETÚBAL
ESCOLA SUPERIOR POLITÉCNICO SETÚBAL

BERNARDO TINOCO FORTALEZA

ALGORIDDLE: Artefacto educacional interativo para apoiar o ensino e aprendizagem de algoritmos.

Relatório de Dissertação de investigação do Mestrado em Engenharia de *Software*

JÚRI

Presidente*:* (Prof. José Cordeiro, ESTSetúbal/IPS)

Orientador*:* (Ph.D., Fábio Ferrentini Sampaio, ESTSetúbal/IPS)

Arguente*:* (Profª. Alcina Prata, ESTSetúbal/IPS)

Data da realização da prova (Dezembro 2025)

Dedico esta dissertação aos meus pais, pelo apoio e incentivo ao longo de todo o meu percurso académico, e a todos os docentes e colegas que partilharam comigo o interesse pelo mundo da computação.

Resumo

Esta dissertação apresenta o *Algoriddle*, um ambiente virtual interativo desenvolvido para apoiar o ensino e a aprendizagem de algoritmos através de visualização, manipulação e simulação em tempo real. A natureza abstrata dos algoritmos dificulta a compreensão dos processos lógicos, especialmente para estudantes iniciantes. O *Algoriddle* foi concebido para responder a esse desafio, integrando princípios pedagógicos e tecnológicos que promovem aprendizagem ativa, pensamento computacional e raciocínio lógico.

A investigação seguiu o modelo DSR Educacional, abrangendo a identificação do problema, o levantamento de requisitos, a conceção, a implementação e a avaliação do artefacto. O sistema permite explorar de forma dinâmica alguns dos algoritmos lecionados nas unidades curriculares de algoritmos e estruturas de dados, selecionados com base no levantamento dos programas das principais universidades de Portugal.

O desenvolvimento recorreu a tecnologias como *React*, *TypeScript*, *Node.js* e *Express*, incluindo um módulo de execução controlada de código baseado na *Piston API* e mecanismos de segurança, registo e monitorização. A avaliação, realizada com estudantes e profissionais, utilizou a *System Usability Scale* (SUS) e escalas Likert, revelando perceções positivas de usabilidade, clareza e motivação. Embora não avaliado em contexto de sala de aula, o sistema demonstra potencial para apoiar a compreensão de algoritmos, constituindo base para estudos futuros.

Palavras-chave: algoritmos, visualização, ensino, simulação, usabilidade, *Design Science Research*

Abstract

This dissertation presents Algoriddle, an interactive virtual environment designed to support the teaching and learning of algorithms through real-time visualization, manipulation, and simulation. The abstract nature of algorithms often makes it difficult for beginners to understand the underlying logical processes. Algoriddle was conceived to address this challenge by integrating pedagogical and technological principles that foster active learning, computational thinking, and logical reasoning.

The research followed the Educational DSR model, covering problem identification, requirements gathering, design, implementation, and evaluation of the artefact. The system enables dynamic exploration of selected algorithms commonly taught in courses on algorithms and data structures, chosen based on a survey of curricula from major Portuguese universities.

The development relied on technologies such as React, TypeScript, Node.js and Express, and includes a controlled code-execution module based on the Piston API, complemented by security, logging, and monitoring mechanisms. The evaluation, conducted with students and professionals, used the System Usability Scale (SUS) and Likert scales, revealing positive perceptions of usability, clarity, and motivation. Although not tested in a classroom setting, the system shows potential to support the understanding of algorithms and provides a foundation for future studies.

Keywords: algorithms, visualization, teaching, simulation, usability, Design Science Research

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Acrónimos

ACM	*Association for Computing Machinery*
ADT	*Abstract Data Type*
API	*Application Programming Interface*
BFS	*Breadth-First Search*
BPMN	*Business Process Model and Notation*
CA	*Constructive Approach*
CPU	*Central Processing Unit*
DFS	*Depth-First Search*
DSR	*Design Science Research*
DSR-E	*Design Science Research* Educacional
EPD	Encarregado de Proteção de Dados
FAQs	*Frequently Asked Questions*
IA	Inteligência Artificial
IEEE	*Institute of Electrical and Electronics Engineers*
IPB	Instituto Politécnico de Bragança
IPBeja	Instituto Politécnico de Beja
IPC	Instituto Politécnico de Coimbra
IPCA	Instituto Politécnico do Cavado e do Ave
IPCB	Instituto Politécnico de Castelo Branco
IPG	Instituto Politécnico da Guarda
IPL	Instituto Politécnico de Lisboa

IPLeiria	Instituto Politécnico de Leiria
IPP	Instituto Politécnico do Porto
IPPt	Instituto Politécnico de Portalegre
IPS	Instituto Politécnico de Setúbal
IPSantarém	Instituto Politécnico de Santarém
IPT	Instituto Politécnico de Tomar
IPV	Instituto Politécnico de Viseu
IPVC	Instituto Politécnico de Viana do Castelo
ISCTE	Instituto Universitário de Lisboa
IST	Instituto Superior Técnico
ISO	*International Organization for Standardization*
JSON	*JavaScript Object Notation*
LMS	*Learning Management System*
MVC	*Model-View-Controller*
NOVA	Universidade Nova de Lisboa
RAM	*Random Access Memory*
RGPD	Regulamento Geral de Proteção de Dados
STL	*Standard Template Library*
SUS	*System Usability Scale*
UA	Universidade de Aveiro
UAc	Universidade dos Açores
UAlg	Universidade do Algarve
UBI	Universidade da Beira Interior

UC	Universidade de Coimbra
UDL	*Universal Design for Learning*
UEvora	Universidade de Évora
UM	Universidade do Minho
UMa	Universidade da Madeira
ULisboa	Universidade de Lisboa
UPorto	*Universidade do Porto*
W3C	*World Wide Web Consortium*
WCAG	*Web Content Accessibility Guidelines*
XP	*Extreme Programming*

Glossário

[A]

- **Acessibilidade** – Capacidade de um sistema, aplicação ou conteúdo digital ser utilizado por pessoas com diferentes necessidades, incluindo deficiências visuais, auditivas, motoras ou cognitivas.
- **ADT *map*** – Tipo de dado abstrato que associa chaves a valores, permitindo operações de inserção, pesquisa e remoção de forma eficiente.
- ***Algoriddle*** – Artefacto educacional desenvolvido nesta investigação, concebido como um ambiente virtual interativo destinado ao ensino e aprendizagem de algoritmos.
- **Algoritmo** – Sequência finita e ordenada de instruções que descreve a resolução de um problema ou a execução de uma tarefa de forma sistemática.
- **Algoritmos de aproximação** – Métodos que produzem soluções próximas da ótima para problemas complexos (como os NP-completos).
- **Algoritmos determinísticos e não determinísticos** – Algoritmos que, respetivamente, produzem sempre o mesmo resultado para uma mesma entrada, ou podem gerar resultados diferentes devido a escolhas internas.
- **Algoritmos e estruturas de dados** – Área fundamental da computação que estuda métodos de processamento e organização eficiente da informação.
- **Algoritmos gananciosos** – Paradigma de design de algoritmos que escolhe localmente a melhor opção em cada passo, visando uma solução global ótima.
- **Algoritmos de ordenação** – Conjunto de algoritmos utilizados para organizar elementos de uma lista segundo uma determinada ordem, como *bubble sort*, *insertion sort, merge sort* ou *quick sort*.
- **Algoritmos de pesquisa** – Técnicas que permitem localizar um elemento específico dentro de uma coleção de dados, como a pesquisa sequencial e a pesquisa binária.
- **Algoritmos recursivos** – Algoritmos que resolvem um problema chamando-se a si próprios em subproblemas menores até atingir um caso base, sendo amplamente usados em árvores, grafos e ordenações.
- **Algoritmos sobre cadeias de *strings*** – Conjunto de algoritmos usados para pesquisa, correspondência, comparação e manipulação de cadeias de texto.

- **Algoritmos de travessia** – Conjunto de algoritmos usados para visitar todos os nós de uma estrutura de dados.
- ***Algorithm Visualizer*** – Ferramenta de visualização interativa que permite observar passo a passo a execução de algoritmos, mostrando operações, comparações e alterações em estruturas de dados, facilitando a compreensão do seu funcionamento.
- **Análise amortizada** – Técnica de análise que avalia o custo médio de operações sucessivas num algoritmo, considerando picos de custo.
- **Análise assintótica** – Estudo do comportamento de um algoritmo para entradas de tamanho muito grande, usando notações como *Big-O, Ω e Θ*.
- **API** – *Interface* de programação de aplicações que permite a comunicação entre diferentes sistemas de *software* através de um conjunto de métodos e protocolos.
- **Aprendizagem ativa** – Abordagem pedagógica que incentiva o envolvimento direto do estudante no processo de aprendizagem, promovendo a experimentação, exploração e resolução de problemas.
- **Aprendizagem passiva** – Modelo de ensino centrado na transmissão de conhecimento, em que o estudante assume um papel predominantemente recetivo.
- **Arquitetura do sistema** – Estrutura organizacional de um sistema de *software*, definindo os seus componentes, funcionalidades e interações.
- **Arquitetura modular** – Modelo de desenvolvimento que organiza um sistema em módulos independentes e reutilizáveis, facilitando a manutenção e a escalabilidade.
- **Artefacto** – Produto tecnológico ou conceptual criado com o propósito de resolver um problema ou demonstrar um princípio teórico no âmbito da investigação.
- **Artefacto educacional** – Artefacto concebido especificamente para fins pedagógicos, visando apoiar o processo de ensino e aprendizagem.
- **Árvores –** Estruturas de dados hierárquicas formadas por nós e ligações, usadas para representar relações pai-filho.
- **Árvore 2-3 –** Árvore de pesquisa balanceada em que cada nó interno pode ter dois ou três filhos e um ou dois valores, garantindo que todas as folhas estão à mesma profundidade e permitindo inserções, remoções e pesquisas eficientes.
- **Árvore AVL –** Tipo de árvore binária de pesquisa equilibrada em que a diferença de altura entre subárvores é no máximo 1.

- **Árvore B –** Estrutura de dados em árvore balanceada utilizada principalmente em bases de dados e sistemas de ficheiros. Permite inserções, remoções e pesquisas eficientes, mantendo os dados ordenados e equilibrados.
- **Árvore binária de pesquisa –** Árvores em que cada nó possui no máximo dois filhos e onde o valor à esquerda é menor e o valor à direita é maior que o do nó pai, permitindo pesquisas, inserções e eliminações ordenadas.
- **Árvore geradora mínima** – Subconjunto de arestas de um grafo ponderado que liga todos os vértices com o menor custo total possível, sem formar ciclos.
- **Árvores de decisão** – Estruturas hierárquicas usadas para tomar decisões ou classificar dados, em que cada nó interno representa uma condição ou teste, e cada ramo representa uma possível decisão ou resultado.
- **Árvore genérica** – Estrutura hierárquica em que cada nó pode ter qualquer número de filhos, usada para representar dados organizados em níveis sem restrições de ramificação.
- **Árvore de Huffman** – Estruturas utilizadas em compressão de dados, nas quais caracteres mais frequentes têm códigos binários mais curtos, minimizando o tamanho total da codificação.
- **Árvore de intervalos** – Estruturas de dados em árvore utilizadas para armazenar intervalos e permitir consultas rápidas, como apagar os intervalos que se sobrepõem a um dado ponto ou intervalo.
- **Árvore n-área (**– Estruturas de dados em que cada nó pode ter até *n* filhos, generalizando o conceito de árvore binária.
- **Árvore de pesquisa balanceada –** Estruturas de dados em árvore que mantêm o equilíbrio entre os ramos após inserções ou remoções, garantindo tempos de pesquisa, inserção e eliminação eficientes.
- **Árvore de prefixos –** Estrutura de dados em árvore usada para armazenar cadeias de caracteres de forma hierárquica, partilhando prefixos comuns entre palavras.
- **Árvore *red-black* –** Tipo de árvore binária de pesquisa autoequilibrada que usa cores (vermelho e preto) para garantir equilíbrio.
- **Árvore *splay* –** Árvore binária autoajustável que move o nó acedido para a raiz através de rotações, melhorando o desempenho em acessos repetidos.
- **Árvore de sufixos –** Estrutura de dados que representa todos os sufixos de uma cadeia de caracteres, permitindo pesquisas e correspondências rápidas em tempo linear.

- **Árvore VP –** Estrutura de dados usada para pesquisa eficiente em espaços métricos, baseada em distâncias relativas entre pontos, útil em aplicações como reconhecimento de padrões e *machine learning*.
- ***Array* –** Estrutura de dados linear que armazena elementos do mesmo tipo em posições contíguas de memória.
- ***Array* multidimensional –** Estrutura de dados que armazena elementos do mesmo tipo em várias dimensões. Pode ser visto como uma lista de listas, onde cada lista dentro da principal tem seus próprios elementos. É usado, por exemplo, para representar uma matriz com várias linhas e colunas.
- ***Array* de sufixos –** Estrutura de dados baseada em *arrays* ordenados de índices que representam todos os sufixos de uma *string*, oferecendo uma alternativa mais compacta às árvores de sufixos.
- **Avaliação empírica** – Etapa da metodologia DSR que consiste na observação e análise sistemática do desempenho de um artefacto em contextos reais.
- **Avaliação de expressões infixas e pós-fixas –** Processo de cálculo de expressões matemáticas representadas em notação infixa (com operadores entre operandos) ou pós-fixa (operadores após operandos), comum em interpretadores e compiladores.

[B]

- ***Backend*** – Camada de um sistema responsável pelo processamento lógico, gestão de dados e comunicação com o servidor, não visível ao utilizador final.
- ***Backups* –** Cópias de segurança de dados ou sistemas, criadas para prevenir perda de informação em caso de falha, erro ou desastre.
- ***Bags* –** Estrutura de dados que armazena elementos sem impor uma ordem específica e permitindo duplicações. Utilizada para contar elementos ou armazenar coleções onde a ordem não é relevante.
- **Bases de dados** – Conjuntos organizados de informação, estruturados para armazenamento, consulta e atualização eficientes.
- ***Bubble sort*** – Algoritmo de ordenação simples que compara pares de elementos adjacentes e os troca até que a lista esteja ordenada.
- ***Bucket sort*** – Algoritmo de ordenação que distribui elementos em baldes (*buckets*) e ordena cada grupo individualmente.
- ***Bundler*** - Ferramenta que agrupa vários arquivos de código em um ou mais arquivos menores, para otimizar o desempenho e reduzir o número de requisições ao servidor.

- ***Boyer-Moore*** – Algoritmo eficiente de pesquisa de padrões em cadeias de caracteres, usado em processamento de texto.
- ***Btree-js –*** Ferramenta para visualização interativa de árvores B, utilizada no ensino de estruturas de dados e bases de dados.

[C]

- **C** – Linguagem de programação de propósito geral, estruturada e eficiente, amplamente utilizada no desenvolvimento de sistemas e *software* de baixo nível.
- **C++** – Linguagem de programação de alto desempenho que suporta paradigmas imperativo, genérico e orientado a objetos.
- **Caminho mais curto (*shortest path*)** – Problema clássico em grafos que procura a rota de custo mínimo entre dois nós.
- ***Chatbot*** – Aplicação que utiliza linguagem natural para interagir com utilizadores, respondendo a perguntas ou executando tarefas automatizadas.
- ***ChatGPT*** – Modelo de linguagem baseado em inteligência artificial desenvolvido pela *OpenAI*, projetado para gerar respostas em linguagem natural e realizar tarefas de processamento de linguagem, como conversação, resumo de texto, tradução e análise de sentimentos, entre outras.
- ***Chrome*** – Navegador *web* desenvolvido pelo *Google*, utilizado para aceder à *internet* e executar aplicações *web*.
- **Ciências da Computação** – Campo científico que estuda os fundamentos teóricos, metodológicos e práticos do processamento de informação por meio de sistemas computacionais.
- **Ciências descritivas** – Área da ciência que procura compreender e explicar fenómenos existentes, como factos ou eventos que já ocorrem na natureza ou na sociedade. O foco está em descrever o que é, sem necessariamente criar algo.
- **Ciências do artificial** – Área da ciência que se dedica ao *design* e criação de soluções ou artefactos para resolver problemas concretos, muitas vezes envolvendo a construção de sistemas ou tecnologias que não existem naturalmente.
- **Ciclo da relevância** – Componente da DSR que assegura a ligação entre o problema real e a solução proposta, garantindo a aplicabilidade prática do artefacto.
- **Ciclo do *design*** – Etapa iterativa da DSR dedicada à criação, desenvolvimento e refinamento de artefactos.

- **Ciclo do rigor** – Dimensão da DSR que assegura o uso adequado de teorias, métodos e conhecimentos científicos durante o processo de *design*.
- **Comissão de ética institucional** – Órgão responsável por avaliar e aprovar projetos de investigação, garantindo que estão de acordo com princípios éticos e legais, especialmente na manipulação de dados de participantes.
- **Compatibilidade** – Capacidade de um sistema funcionar corretamente em diferentes ambientes ou plataformas.
- **Complexidade assintótica** – Estudo do comportamento do tempo ou espaço de execução de um algoritmo à medida que o tamanho da entrada tende para o infinito, expresso em notações como *Big-O*, Ω e Θ.
- **Complexidade computacional** – Área da computação que analisa o desempenho e os recursos necessários à execução de algoritmos, como tempo e memória.
- **Complexidade espacial –** Medição que avalia o espaço de memória que um algoritmo necessita durante a sua execução, incluindo variáveis, estruturas de dados e memória auxiliar.
- **Complexidade temporal –** Medição que avalia o tempo de execução de um algoritmo em função do tamanho do *input*. Permite estimar a rapidez com que um algoritmo processa dados.
- **Componentes** – Unidades independentes de *software* que encapsulam funcionalidades específicas dentro de um sistema maior.
- **Concepção do artefacto** – Fase do processo de *design* que envolve a planificação, modelação e implementação inicial do artefacto.
- **Conjecturas comportamentais** – Hipóteses que explicam os efeitos esperados do uso de um artefacto sobre o comportamento ou desempenho dos utilizadores.
- **Conjuntos de *strings*** – Estruturas de dados que armazenam coleções de cadeias de caracteres, permitindo operações de inserção, remoção e pesquisa eficientes, muitas vezes implementadas com árvores de prefixos.
- **Conjuntos dinâmicos** – Estruturas de dados que permitem inserção, remoção e pesquisa de elementos ao longo da execução do programa.
- **Conjuntos disjuntos** – Estrutura de dados que mantém uma coleção de conjuntos não sobrepostos, suportando operações de união e pesquisa eficiente, frequentemente usada em algoritmos de grafos.
- ***Constructive Approach*** – Abordagem metodológica de caráter construtivo, centrada na criação de soluções tecnológicas inovadoras para problemas específicos.

- **Construcionismo** – Abordagem educacional que amplia o construtivismo, enfatizando a construção de artefactos tangíveis como parte do processo de aprendizagem, como criar projetos ou resolver problemas práticos.
- **Construtivismo** – Teoria da aprendizagem que defende que os indivíduos constroem o seu próprio conhecimento de forma ativa, a partir das suas experiências e interações com o ambiente.
- ***Counting sort*** – Algoritmo de ordenação estável que utiliza contagem de ocorrências dos valores para ordenar listas de números inteiros de forma eficiente.
- **CPU** – Unidade central de processamento responsável por executar instruções de programas.
- **Critérios de eficiência** – Conjunto de parâmetros usados para avaliar o desempenho de algoritmos, como tempo, espaço e escalabilidade.
- ***CS Unplugged*** – Conjunto de atividades educativas que ensinam conceitos de computação sem o uso de computadores.
- **Custo mínimo** – Conceito de otimização em grafos, relacionado a encontrar o menor custo total entre vértices.

[D]

- **Dados sociodemográficos** – Informações que descrevem características sociais e demográficas dos participantes numa investigação, como idade, género ou formação.
- ***Data Structure Visualizations*** – Desenvolvido pela Universidade de São Francisco, foca-se na demonstração de operações fundamentais sobre estruturas de dados, como pilhas, filas, árvores e listas, permitindo observar de forma clara como estas se reorganizam durante operações de inserção ou remoção.
- **Dicionários –** Estrutura de dados que armazenam pares de chave-valor, permitindo inserções, remoções e pesquisas eficientes, frequentemente implementada com tabelas de dispersão ou árvores de pesquisa.
- **Deques** – Estrutura de dados que permite inserções e remoções em ambas as extremidades (início e fim), combinando características de pilhas e filas.
- ***Design*** – Processo de concepção que combina aspetos estéticos, funcionais e técnicos para criar soluções eficazes e centradas no utilizador.
- ***Design Science Research*** – Metodologia de investigação que combina rigor científico e relevância prática por meio da criação e avaliação de artefactos inovadores.

- **Diagramas de casos de uso** – Representações gráficas das interações entre utilizadores e um sistema.
- **Diagramas de sequência –** Diagramas UML que mostram a ordem temporal das interações entre componentes de um sistema.
- **Diminuir e conquistar –** Paradigma de design de algoritmos que resolve um problema reduzindo progressivamente o tamanho da instância.
- **Diretrizes –** Conjunto de recomendações ou normas que orientam a concepção, implementação e avaliação de sistemas, artefactos ou processos, garantindo qualidade, acessibilidade e conformidade com boas práticas.
- **Dispersão dupla** – Técnica de endereçamento aberto que utiliza uma segunda função de *hash* para calcular o intervalo entre tentativas de inserção, reduzindo colisões e agrupamentos.
- **Dividir para conquistar –** Técnica que resolve problemas dividindo-os em subproblemas menores, resolvendo-os e combinando os resultados.

[E]

- ***Edge*** – Navegador *web* desenvolvido pela *Microsoft*, utilizado para aceder à *internet* e executar aplicações *web*.
- ***E-learning*** – Modalidade de ensino e aprendizagem mediada por tecnologias digitais, que permite acesso remoto a conteúdos e atividades educativas.
- **Eficiência computacional** – Grau em que um algoritmo utiliza de forma otimizada recursos como tempo e memória.
- **Emparelhamento estável** – Problema que visa emparelhar dois conjuntos de elementos de modo que não existam pares que preferissem estar juntos fora da solução.
- **Encadeamento separado** – Técnica para resolver colisões em tabelas de dispersão, armazenando os elementos que partilham o mesmo valor de *hash* em listas ligadas independentes.
- **Encarregado de proteção de dados** – Profissional responsável por garantir a conformidade de uma instituição com as normas de proteção de dados pessoais.
- **Endereçamento aberto** – Técnica usada em tabelas de dispersão (*hash tables*) para resolver colisões, procurando outra posição livre dentro da própria tabela segundo uma sequência de sondagem.

- **Endereçamento linear –** Tipo de endereçamento aberto em que, após uma colisão, a procura por uma posição livre é feita de forma sequencial (por exemplo, posição seguinte, depois a próxima, e assim sucessivamente).
- **Endereçamento quadrático –** Variante do endereçamento aberto em que a sequência de sondagem aumenta quadraticamente (por exemplo, 1^2, 2^2, 3^2...), reduzindo agrupamentos de colisões em comparação com o endereçamento linear.
- **Equações de recorrência –** Expressões matemáticas que definem o tempo de execução de um algoritmo recursivo em função de subproblemas menores.
- **Escala Likert –** Instrumento de avaliação que mede atitudes ou perceções dos utilizadores com base em níveis de concordância.
- **Escalabilidade** – Capacidade de um sistema crescer em desempenho e capacidade sem comprometer a eficiência ou a estabilidade.
- **Escalabilidade Horizontal** – Capacidade de um sistema ou aplicação aumentar a sua performance e capacidade ao adicionar mais unidades de *hardware* ou nós em paralelo, em vez de melhorar a potência de um único componente.
- **Estado da arte** – Análise crítica das soluções, métodos e teorias existentes sobre um tema de investigação.
- **Estruturas de dados** – Modelos de organização de informação que permitem armazenar, aceder e manipular dados de forma eficiente.
- **Estruturas de dados lineares e hierárquicas** – Classificação das estruturas de dados: lineares (como listas e filas), onde os elementos têm uma sequência única, e hierárquicas (como árvores e grafos), onde existem relações de níveis ou dependência.
- **Estruturas de dados multidimensionais** – Estruturas que organizam dados em mais de uma dimensão, como matrizes, *quadtrees* ou *k-d trees*, usadas em aplicações gráficas e científicas.
- **Estruturas lineares** – Conjunto de estruturas de dados como listas, pilhas e filas, organizadas de forma sequencial.
- ***Express*** – *Framework* para *Node.js* que facilita o desenvolvimento de aplicações *web* e APIs, simplificando a gestão de rotas e pedidos HTTP.

[F]

- ***FAQs*** – Secção com respostas a perguntas comuns sobre um produto ou serviço.
- ***Feedback*** – Retorno fornecido ao utilizador ou aprendiz sobre o desempenho ou resultado de uma ação, essencial para o processo de aprendizagem.

- **Ferramentas de visualização** – Aplicações que permitem representar graficamente dados, conceitos ou processos, facilitando a compreensão e o raciocínio.
- **Fiabilidade** – Grau em que um sistema ou instrumento produz resultados consistentes e livres de erros sob as mesmas condições.
- **Filas** – Estruturas de dados que seguem o princípio FIFO (*First In, First Out*).
- **Filas de prioridade** – Estruturas em que cada elemento tem uma prioridade associada, determinando a ordem de processamento.
- ***Firefox*** – Navegador *web* desenvolvido pela *Mozilla*, utilizado para aceder à *internet* e executar aplicações *web*.
- **Fluxos de dados** – Representação do movimento e transformação de dados através de um sistema ou programa, indicando como as informações entram, são processadas e produzem *outputs*.
- **Fluxogramas** – Diagramas gráficos que representam o fluxo de execução de um algoritmo ou processo, facilitando a visualização de decisões e iterações.
- **Fluxo máximo** – Problema em teoria dos grafos que determina o maior fluxo possível entre um vértice de origem e um vértice de destino, respeitando as capacidades das arestas.
- ***Framework*** – Conjunto estruturado de ferramentas e bibliotecas que fornece uma base para o desenvolvimento de *software*.
- ***Frontend*** – Camada de uma aplicação responsável pela *interface* e interação direta com o utilizador.
- **Funções de referência** – Funções que servem de base para comparar o crescimento assintótico de outras funções em análises de complexidade.
- ***Funções hash*** – Funções que convertem dados em valores numéricos usados para indexação em tabelas de dispersão (*hash tables*).

[G]

- **Genéricos** – Conceito de programação que permite criar funções e classes que operam com diferentes tipos de dados, promovendo reutilização e segurança de tipo.
- ***Geogebra*** – *Software* educativo que combina geometria, álgebra e cálculo, amplamente utilizado no ensino de conceitos matemáticos e científicos.
- ***GitHub*** – Plataforma de hospedagem de código e colaboração baseada em *Git*.
- ***Google Drive*** – Serviço de armazenamento na nuvem que permite guardar, partilhar e sincronizar ficheiros *online*.

- **Grafos** – Estruturas de dados compostas por vértices e arestas, usadas para representar relações ou redes.

[H]

- ***Heap* –** Estrutura de dados em forma de árvore usada em filas de prioridade e algoritmos de ordenação como heap sort.
- ***Heap* binário (*binary heap*) –** Estrutura de dados baseada numa árvore binária completa que satisfaz a propriedade de heap: em um max-heap, cada nó é maior ou igual aos seus filhos; em um min-heap, cada nó é menor ou igual aos seus filhos.
- ***Heap* binomial (*binomial heap*)–** Estrutura de dados composta por várias árvores binomiais, usada em operações eficientes de filas de prioridade.
- ***Heap* de Fibonacci (*Fibonacci heap*) –** Estrutura de dados avançada que otimiza operações de inserção e diminuição de chave em filas de prioridade.
- ***Heap sort* –** Algoritmo de ordenação baseado em uma estrutura de dados do tipo heap, que constrói uma heap a partir dos elementos e extrai repetidamente o maior (ou menor) elemento.
- **Heurísticas –** Estratégias aproximadas que encontram soluções aceitáveis para problemas complexos quando não é viável obter a solução ótima.

[I]

- **Identificação do problema** – Primeira etapa da DSR, que consiste em compreender e definir claramente o problema que motiva o desenvolvimento do artefacto.
- **IEEE** – Organização internacional que define padrões e normas em engenharia, computação e tecnologias relacionadas.
- **Implementações vetoriais e dinâmicas** – Formas de representar estruturas de dados: as vetoriais usam *arrays* de tamanho fixo; as dinâmicas ajustam-se automaticamente à quantidade de elementos, permitindo inserções e remoções flexíveis.
- **Infraestrutura** – Conjunto de recursos físicos e lógicos (*hardware*, *software*, redes, servidores) necessários para suportar e operar sistemas, aplicações e serviços de forma eficiente e segura.
- ***Input*** – Dados ou informações fornecidas a um programa, função ou sistema para que ele realize operações ou produza *outputs*. Em programação, podem ser números, textos, arquivos ou outros tipos de dados.
- **Insertion sort** – Algoritmo de ordenação que constrói a lista ordenada inserindo um elemento de cada vez na posição correta.

- ***Insights*** – Conhecimentos ou perceções adquiridas através da análise de dados, experiências ou investigação, que ajudam a orientar decisões ou melhorias em sistemas e projetos.
- **Inteligência Artificial** – Área da ciência da computação que estuda e desenvolve sistemas capazes de realizar tarefas que normalmente requerem inteligência humana, como reconhecimento de padrões, tomada de decisão e aprendizagem.
- ***Interface* do utilizador** – Elemento visual e interativo de um sistema com o qual o utilizador estabelece contacto direto.
- **Interação homem-máquina** – Refere-se ao estudo e desenvolvimento de sistemas que permitem a comunicação entre seres humanos e computadores.
- **ISO** – Organização internacional responsável pela definição de normas e padrões técnicos que asseguram qualidade, segurança, interoperabilidade e boas práticas em diferentes áreas, incluindo tecnologias de informação.
- **ISO 9241-210:2019** – Norma internacional que define princípios de *design* centrado no utilizador para sistemas interativos, focando-se na ergonomia, na usabilidade e na experiência do utilizador.
- **Iteradores** – Objetos ou mecanismos que permitem percorrer sequencialmente os elementos de uma estrutura de dados sem expor a sua implementação interna.

[J]

- ***Java*** – Linguagem de programação orientada a objetos, multiplataforma e fortemente tipada.
- ***JavaScript*** – Linguagem de programação amplamente usada no desenvolvimento *web* para adicionar interatividade e dinamismo a páginas e aplicações.
- ***Jeliot* 2000** – Ferramenta educacional que visualiza a execução de programas em *Java* de forma gráfica, ajudando os alunos a entenderem a ligação entre o código, a lógica do programa e o comportamento observado durante a execução.

[K]

- **KMP –** Algoritmo de pesquisa de padrões em texto que utiliza informação pré-processada do padrão para evitar recomparações desnecessárias, garantindo eficiência linear.

[L]

- ***Learning Analytics*** – Processo de recolha, análise e interpretação de dados sobre os estudantes e as suas interações, com o objetivo de melhorar o ensino e a aprendizagem.

- **Levantamento de requisitos** – Etapa da análise de sistemas que identifica as necessidades funcionais e não funcionais de um projeto.
- **Linguagem de programação** – Conjunto de regras e sintaxe que permite ao programador escrever instruções compreendidas e executadas por um computador.
- **Listas** – Estrutura de dados linear que armazena uma coleção de elementos em sequência. Permite acesso, inserção e remoção de elementos, geralmente mantendo a ordem de inserção.
- **Listas ligadas** – Estrutura de dados linear composta por nós, em que cada nó contém um valor e uma referência para o nó seguinte. Permite inserções e remoções eficientes, mas o acesso a elementos é sequencial.
- **Listas / matrizes de adjacência** – Representações de grafos em estruturas de dados, onde listas são usadas para armazenamento eficiente de vizinhos e matrizes para acesso rápido às ligações.
- ***Logo*** – Linguagem de programação educativa criada para ensinar conceitos de lógica e programação, famosa pelo controlo de uma "tartaruga" gráfica para desenhar formas geométricas.
- ***Logs*** – Registos detalhados das operações e eventos ocorridos num sistema ou aplicação, usados para monitorização, depuração e auditoria. Normalmente contêm mensagens de erro, alertas e informações sobre o estado do sistema.
- **LSD** – Estratégia de ordenação que processa primeiro os dígitos menos significativos, comum em versões práticas do *radix sort*.

[M]

- ***Machine learning*** – Subcampo da inteligência artificial que permite a sistemas aprenderem automaticamente a partir de dados, identificando padrões e fazendo previsões sem programação explícita para cada tarefa.
- **Manual do utilizador** – Documento que descreve funcionalidades, instruções de utilização e procedimentos de um sistema ou *software*, orientando os utilizadores na operação correta e eficiente do artefacto.
- **Mapeamento de *strings*** – Processo de converter cadeias de caracteres em representações numéricas (*hashes*) ou índices, facilitando pesquisa e comparação em estruturas de dados.
- **Memória dinâmica** – Alocação de espaço de memória durante a execução de um programa, permitindo maior flexibilidade na gestão e utilização eficiente dos recursos.

- **Memorização** – Técnica de otimização que armazena resultados de cálculos anteriores para evitar recomputações em chamadas recursivas, melhorando a eficiência.
- **Melhor, pior e caso médio** – Classificação da eficiência de algoritmos com base no desempenho em diferentes cenários: melhor caso, situação em que o algoritmo executa o mínimo possível de operações; pior caso, situação em que o algoritmo executa o máximo de operações; e caso médio, análise probabilística do desempenho esperado considerando *inputs* variados.
- ***Merge sort*** – Algoritmo de ordenação baseado na estratégia "dividir para conquistar", que divide a lista em sublistas, ordena cada sublista recursivamente e combina as sublistas ordenadas.
- **Metodologia** – Conjunto estruturado de procedimentos, técnicas e princípios utilizados na condução de uma investigação científica.
- **Metodologias iterativas** - Abordagens que envolvem ciclos repetitivos de desenvolvimento, teste e melhoria.
- **Modelagem funcional** – Técnica de representação de sistemas ou processos através de funções e fluxos de dados, focando-se nas operações realizadas, *inputs*, *outputs* e relações entre componentes, sem detalhar a implementação física.
- **Modelação matemática** – Processo de representação de fenómenos do mundo real através de expressões matemáticas e computacionais.
- **Modelo DSR Educacional** – Adaptação da metodologia *Design Science Research* aplicada ao contexto educativo, visando a criação de artefactos pedagógicos.
- ***Modellus*** – *Software* de modelação e simulação matemática usado para estudar fenómenos científicos e educativos, permitindo criar modelos, realizar simulações e analisar resultados.
- **Modelo de Hevner** – Modelo de referência para DSR que define ciclos de rigor e relevância na criação de artefactos tecnológicos.
- **Motivação** – Conjunto de fatores internos e externos que influenciam o envolvimento e o desempenho dos estudantes.
- ***Moodle*** – Plataforma de gestão de aprendizagem (LMS) de código aberto amplamente utilizada em contextos educativos.
- ***MoSCoW*** – Técnica que classifica os requisitos em "*Must have*" (essenciais para o sucesso do projeto), "*Should have*" (importantes, mas não cruciais), "*Could have*" (desejáveis, mas não essenciais) e "*Won't have*" (não serão incluídos nesta fase).

- **MSD** – Estratégia de ordenação usada em algoritmos como o *radix sort*, que processa primeiro os dígitos mais significativos.
- **MVC** – Padrão arquitetónico que separa a lógica de negócio (*Model*), a interface de utilizador (*View*) e o controlo das interações (*Controller*), promovendo modularidade.

[N]

- ***Node.js*** – Ambiente de execução JavaScript orientado a eventos, utilizado para desenvolver aplicações de servidor escaláveis.
- **Notação de complexidade** – Sistema de representação (como Big-O, Ω e Θ) usado para expressar o crescimento do tempo ou espaço requerido por um algoritmo em função do tamanho da entrada.
- **Notação *Big-O*** – Forma matemática de expressar a complexidade de um algoritmo, representando o crescimento do tempo de execução em função do tamanho do *input*.

[O]

- **Ordenação topológica –** Técnica usada em grafos direcionados acíclicos que produz uma sequência linear dos vértices respeitando as dependências entre eles.
- ***Output* –** Resultados ou informações produzidas por um programa, função ou sistema após o processamento dos dados de *input.* Podem ser valores numéricos, textos, gráficos, ficheiros ou outros tipos de dados.

[P]

- **Paradigmas de resolução de problemas** – Abordagens gerais utilizadas na construção de algoritmos, como dividir para conquistar, programação dinâmica ou algoritmos gananciosos.
- **Pensamento computacional** – Capacidade de formular problemas e soluções de modo que possam ser executados de forma eficiente por um computador.
- **Pensamento crítico** – Habilidade de analisar, avaliar e interpretar informações de forma lógica e fundamentada para resolver problemas complexos.
- **Percurso em grafos** – Processo de visitar todos os vértices de um grafo através de algoritmos como a procura em profundidade (DFS) ou procura em largura (BFS).
- **Pertinência pedagógica** – Adequação de um recurso educativo aos objetivos de ensino e aprendizagem estabelecidos.
- **Pesquisa binária** - Algoritmo eficiente que divide recursivamente o espaço de pesquisa pela metade até encontrar o elemento desejado.

- **Pesquisa sequencial** - Algoritmo simples que percorre todos os elementos de uma lista até encontrar o valor procurado.
- ***PhET Interactive Simulations*** – Conjunto de simulações interativas para o ensino de ciências, desenvolvidas pela Universidade do Colorado Boulder.
- **Pilhas** – Estrutura de dados que segue o princípio LIFO (Last In, First Out), em que o último elemento inserido é o primeiro a ser removido.
- ***Piston API*** – *Interface* de programação que permite executar código de diversas linguagens de programação de forma remota, geralmente em um ambiente *sandbox* seguro, facilitando testes e execução de *snippets* sem instalar o compilador localmente.
- **Plataforma de gestão de aprendizagem** – Sistema digital que facilita a criação, distribuição e acompanhamento de cursos e atividades formativas.
- **Ponteiros** – Variáveis que armazenam o endereço de memória de outro valor ou objeto, permitindo manipular dados de forma indireta e dinâmica, fundamentais em linguagens como C e C++.
- **Princípios de aprendizagem multimédia** – Conjunto de princípios pedagógicos que orientam a concepção de materiais de aprendizagem multimédia, como texto, imagens e áudio, para maximizar a compreensão e retenção de informação.
- **Princípio da confidencialidade** – Princípio de segurança que assegura que apenas pessoas ou sistemas autorizados podem aceder a dados sensíveis ou privados, protegendo-os de divulgação não autorizada.
- **Princípios fundamentais de Hevner** – Conjunto de princípios que guiam a investigação em DSR, incluindo relevância prática, rigor científico, criação de artefactos e avaliação empírica.
- **Problemas P, NP e NP-Completos** – Classificação de problemas com base na sua complexidade computacional: P são problemas resolvidos em tempo polinomial; NP são problemas cujas soluções podem ser verificadas em tempo polinomial; e NP-Completos são os mais difíceis dentro de NP, para os quais não se conhece uma solução eficiente.
- **Problema Union-Find** – Estrutura de dados que gere conjuntos disjuntos, permitindo operações eficientes de união e procura.
- **Processos computacionais** – Sequências de operações realizadas por um sistema ou programa para processar dados e gerar resultados, incluindo cálculos, armazenamento e comunicação entre componentes.

- **Procura em profundidade –** Algoritmo de exploração de grafos ou árvores que percorre cada ramo até atingir um nó terminal antes de retroceder, permitindo visitar todos os vértices de forma recursiva ou usando uma pilha.
- **Procura em largura –** Algoritmo de exploração de grafos ou árvores que percorre os nós em camadas, visitando todos os vértices a uma determinada distância antes de passar para a camada seguinte, normalmente usando uma fila.
- **Programação dinâmica** – Técnica de resolução de problemas que consiste em decompor um problema em subproblemas mais simples, resolver cada subproblema uma única vez e armazenar os resultados para evitar cálculos repetidos, aumentando a eficiência.
- **Programação genérica** – Paradigma que permite criar algoritmos e estruturas de dados independentes de tipos específicos, aumentando a reutilização e flexibilidade do código.
- **Programação imperativa** – Paradigma de programação baseado em instruções sequenciais que modificam o estado do programa.
- **Programação orientada a objetos** – Paradigma que organiza o código em objetos com propriedades e métodos, promovendo modularidade e reutilização.
- ***Probing* linear e quadrático** – Métodos de endereçamento aberto em tabelas de dispersão para resolver colisões, onde a próxima posição livre é encontrada por incrementos lineares ou quadráticos.
- **Prototipagem** – Processo de criação de versões preliminares de um sistema com o objetivo de testar e aperfeiçoar funcionalidades.
- **Pseudocódigo** – Representação simplificada de um algoritmo que descreve a lógica de um programa de forma textual, sem a sintaxe rígida de uma linguagem de programação específica. É usado para planear, explicar e comunicar soluções de forma clara antes de implementar o código real.
- ***Python*** – Linguagem de programação de alto nível conhecida pela sua clareza sintática e ampla aplicação em ciência de dados, *web* e educação.

[Q]

- ***Quick sort*** – Algoritmo de ordenação eficiente que escolhe um pivô, particiona a lista em elementos menores e maiores que o pivô, e ordena recursivamente cada partição.

[R]

- **Raciocínio lógico** – Processo mental de formulação de inferências coerentes e válidas, essencial para o desenvolvimento de algoritmos e resolução de problemas.

- ***Radix sort*** – Algoritmo de ordenação que processa os elementos dígito a dígito, começando do dígito menos significativo (LSD) ou mais significativo (MSD), utilizando uma abordagem estável e eficiente para números inteiros ou *strings*.
- **RAM –** Memória de acesso aleatório do computador utilizada para armazenar temporariamente dados e instruções durante a execução de programas, permitindo acesso rápido e eficiente.
- **Rastreamento** – Técnica de monitorização da execução de um programa, registando a sequência de operações, valores de variáveis e chamadas de funções para análise e depuração.
- **Re-hashing** – Processo de reconstrução de uma tabela de dispersão com uma nova função *hash* ou tamanho, utilizado quando o número de elementos aumenta e o desempenho começa a degradar-se.
- ***React*** – Biblioteca *JavaScript* para construção de interfaces de utilizador baseadas em componentes reutilizáveis.
- **Recursividade** – Técnica em que uma função se chama a si própria para resolver subproblemas menores, repetindo-se até atingir um caso base que termina o processo.
- **Requisitos funcionais** – Características que descrevem o que um sistema deve fazer para satisfazer as necessidades do utilizador.
- **Requisitos não funcionais** – Propriedades que definem como o sistema deve comportar-se, incluindo desempenho, segurança e usabilidade.
- **Resolução de colisões por encadeamento** – Método de tratamento de colisões em tabelas de dispersão, onde cada posição da tabela armazena uma lista ligada de elementos com o mesmo valor de *hash*.
- **Regulamento geral sobre a proteção de dados** – Legislação europeia que regula o tratamento e proteção de dados pessoais.

[S]

- ***Safari*** – Navegador *web* desenvolvido pela *Apple*, utilizado para aceder à *internet* e executar aplicações *web*.
- ***Sandbox*** – Ambiente controlado de execução que permite testar código de forma segura e isolada.
- ***Scrum*** - *Framework* ágil para desenvolvimento de *software* que organiza o trabalho em sprints (ciclos curtos de desenvolvimento).

- ***Selection sort*** - Algoritmo de ordenação simples que percorre repetidamente a lista, seleciona o menor (ou maior) elemento e o coloca na posição correta.
- **Servidor** – Computador ou sistema responsável por fornecer serviços, dados ou aplicações a outros dispositivos numa rede.
- ***Shell sort*** – Algoritmo de ordenação que generaliza o *insertion sort*, comparando elementos distantes por "*gap*" e reduzindo gradualmente esse intervalo, melhorando a eficiência em listas médias a grandes.
- ***Skip Lists*** – Estrutura de dados probabilística que permite pesquisa, inserção e remoção eficientes, combinando listas ligadas com múltiplos níveis de referência para acelerar o acesso a elementos.
- **Sobrecarga de funções** – Técnica que permite definir múltiplas funções com o mesmo nome mas assinaturas diferentes (número ou tipo de parâmetros), facilitando a reutilização e legibilidade do código.
- **STL** – Biblioteca padrão do C++ que fornece estruturas de dados e algoritmos genéricos, como vetores, listas, mapas, conjuntos e iteratores.
- ***System Usability Scale*** – Questionário padronizado utilizado para avaliar a usabilidade percebida de sistemas e aplicações.

[T]

- **Tabelas de dispersão** – Estruturas de dados que mapeiam chaves a valores através de uma função de hash, permitindo acesso, inserção e remoção.
- **Tabelas de símbolos elementares** – Estruturas simples de armazenamento que associam chaves a valores, geralmente implementadas com listas ou arrays ordenados.
- ***Templates*** – Recurso de programação genérica que permite criar funções e classes que operam sobre diferentes tipos de dados sem duplicar código, amplamente usado em C++.
- **Teorema mestre** – Método para resolver equações de recorrência típicas de algoritmos "dividir para conquistar", permitindo determinar a complexidade assintótica de forma direta.
- **Teoria da carga cognitiva de Sweller** – Modelo que explica como a capacidade limitada da memória de trabalho afeta a aprendizagem e o desempenho cognitivo.
- ***Timeouts*** – Limites de tempo definidos para a execução de operações, pedidos ou processos. Quando o tempo expira antes de a operação ser concluída, ocorre uma interrupção automática para evitar bloqueios ou sobrecarga do sistema.

- ***TypeScript*** – *Superset* do *JavaScript* que adiciona tipagem estática e funcionalidades avançadas de desenvolvimento.

[U]

- ***Universal Design for Learning*** – Modelo que explica como a capacidade limitada da memória de trabalho afeta a aprendizagem e o desempenho cognitivo.
- **Usabilidade** – Qualidade de um sistema que mede a facilidade, eficiência e satisfação com que o utilizador o utiliza para atingir objetivos.

[V]

- ***Vite*** – Ferramenta moderna de *build* para projetos *JavaScript* e *TypeScript*, focada em desempenho e simplicidade.
- ***Visualgo*** – Plataforma online de visualização interativa de algoritmos e estruturas de dados, destinada ao ensino.

[W]

- ***Web*** – Qualidade de um sistema que mede a facilidade, eficiência e satisfação com que o utilizador o utiliza para atingir objetivos.
- **W3C/WAI** – Organizações responsáveis pela definição de normas e diretrizes para acessibilidade e interoperabilidade na *web*.
- ***WCAG* 2.1, nível AA** – Norma internacional que define critérios de acessibilidade digital, assegurando que conteúdos *web* sejam utilizáveis por pessoas com diferentes limitações.

[X]

- **XP** – Metodologia ágil que promove práticas como programação em pares, desenvolvimento contínuo e feedback rápido.

1 Introdução

Este capítulo apresenta o tema central da dissertação, dedicado ao ensino e à aprendizagem de algoritmos, estabelecendo o enquadramento geral da investigação. Inicialmente, descreve-se o contexto e a motivação subjacentes ao estudo, seguido pela explicitação dos objetivos que orientam o desenvolvimento do artefacto proposto. Em seguida, introduzem-se as questões de investigação que estruturam o trabalho, e, por fim, detalha-se a organização do documento, destacando as etapas que compõem o desenvolvimento do estudo.

1.1 Contexto e motivação

O ensino e a aprendizagem de algoritmos ocupam um lugar estruturante na formação em Engenharia Informática e Ciências da Computação, contribuindo de forma decisiva para o desenvolvimento do pensamento computacional. (Wing, 2006) descreve este tipo de pensamento como uma competência essencial para a resolução sistemática, lógica e abstrata de problemas. Assim, torna-se importante recorrer a metodologias e ferramentas que tornem mais claro como os algoritmos funcionam, facilitando a sua compreensão.

Não obstante a sua importância, diversos estudos apontam dificuldades persistentes na aprendizagem destes conteúdos, sobretudo devido ao seu elevado nível de abstração. (Grover & Pea, 2013) sublinham que o desenvolvimento do pensamento computacional implica competências cognitivas complexas, muitas vezes ainda em consolidação nas fases iniciais da formação, o que pode comprometer a compreensão de conceitos algorítmicos fundamentais.

Neste contexto, a investigação sobre visualização de algoritmos tem procurado identificar abordagens que apoiem a aprendizagem. Contudo, a evidência empírica não é linear. (Byrne *et al.*, 1999) observaram que as animações podem gerar tendências positivas, embora nem sempre resultem em melhorias estatisticamente significativas quando comparadas com métodos exclusivamente textuais, indicando que a mera inclusão de elementos visuais não garante automaticamente ganhos de aprendizagem.

Estudos posteriores enfatizaram a importância da interatividade. (Hundhausen *et al.*, 2002) demonstraram que a eficácia das visualizações depende do envolvimento ativo do estudante, que deve poder experimentar diferentes passos da execução do algoritmo, observar o que acontece em cada momento e explorar consequências dessas ações. De forma complementar, (Naps *et al.*, 2002) argumentam que níveis superiores de interação tendem a promover maior envolvimento cognitivo.

Estes resultados relacionam-se com as perspetivas construtivistas e construcionistas, como as de (Piaget, 1970) e (Papert, 1980), que defendem que o conhecimento é construído através da ação e da experimentação. No entanto, (Shaffer *et al.*, 2007) relatam que muitas visualizações existentes são de baixa qualidade, oferecem pouco controlo ao utilizador (por exemplo, sobre dados ou ritmo de execução) e carecem de valor pedagógico claro, o que pode limitar a sua utilidade no ensino superior.

Deste modo, torna-se pertinente o desenvolvimento de ambientes que articulem de forma eficaz princípios pedagógicos, soluções tecnológicas e níveis adequados de interatividade. É precisamente neste contexto que surge o *Algoriddle*, um ambiente digital concebido com o objetivo de facilitar a exploração visual e interativa dos algoritmos.

O propósito deste ambiente é permitir que os utilizadores acompanhem passo a passo o que o algoritmo faz e interfiram nesses passos quando necessário, tornando mais acessíveis conceitos que, frequentemente, são apresentados de forma abstrata. Assim, o *Algoriddle* surge como um recurso complementar às práticas pedagógicas já existentes, não se propondo a substituir as metodologias em vigor, mas antes a enriquecer a experiência de ensino e aprendizagem.

Estas funcionalidades, que permitem uma interação mais direta com os algoritmos, serão exploradas em detalhes no Capítulo 4, onde será abordado como podem ser utilizadas para aprofundar o entendimento dos conceitos e otimizar a experiência de aprendizagem.

Embora parte da literatura fundacional citada neste enquadramento remonte ao final da década de 1990 e início dos anos 2000, essas referências continuam a ser amplamente utilizadas por estabelecerem conceitos basilares no domínio da visualização e interatividade de algoritmos. Importa, contudo, reconhecer que o contexto tecnológico atual é substancialmente distinto, marcado por aplicações web ricas, interfaces altamente interativas e maior preocupação com acessibilidade e experiência do utilizador. Nesse sentido, o presente trabalho assume esses contributos clássicos como ponto de partida conceptual, procurando materializá-los num artefacto contemporâneo, desenvolvido com tecnologias atuais e alinhado com as expectativas de interação dos utilizadores atuais.

1.2 Objetivos e resultados esperados

O principal objetivo desta dissertação consiste em conceber, implementar e avaliar um ambiente virtual interativo orientado para o apoio ao ensino e à aprendizagem de algoritmos. Para tal, adotou-se o modelo *Design Science Research* Educacional (DSR-E), proposto por (Pimentel *et al.*, 2020), o qual adapta o *Design Science Research* (DSR) ao contexto educativo. Com base neste enquadramento metodológico, o *Algoriddle* foi desenvolvido com o propósito de oferecer representações visuais, mecanismos de interação e elementos explicativos que potencialmente

possam apoiar a compreensão dos algoritmos, reconhecendo-se que o seu impacto pedagógico direto não foi avaliado em contexto curricular.

Neste âmbito, espera-se, em primeiro lugar, analisar a clareza das visualizações, explicações e funcionalidades de interação disponibilizadas pela aplicação, tendo como referência a perceção dos utilizadores. Em segundo lugar, pretende-se avaliar a usabilidade geral do artefacto com recurso ao *System Usability Scale* (SUS) (Brooke, 1996), complementado pelas respostas aos itens incluídos no questionário de avaliação desenvolvido para este estudo. Por fim, procura-se recolher opiniões sobre a utilidade pedagógica percebida da aplicação, considerando o seu potencial para apoiar a compreensão dos algoritmos abordados.

1.3 Questões de investigação

Tendo em conta o problema identificado, os objetivos definidos e os resultados esperados, estabeleceram-se duas questões de investigação que orientaram o desenvolvimento e a avaliação do *Algoriddle*. A primeira consiste em compreender como os utilizadores percecionam a usabilidade, a clareza e a utilidade pedagógica potencial da aplicação, abrangendo aspetos relacionados com a experiência de utilização, a compreensão dos elementos visuais e a utilidade percebida das funcionalidades disponibilizadas.

A segunda questão procura analisar de que forma o modelo DSR-E estruturou o processo de conceção, desenvolvimento e avaliação do Algoriddle. Esta questão é discutida de forma aprofundada no Capítulo 3, onde se descreve a aplicação concreta das etapas do DSR-E, incluindo a identificação do problema, a definição dos requisitos, o processo de *design* do artefacto, a sua implementação e a fase de avaliação com utilizadores.

Estas duas questões articulam-se diretamente com os objetivos do estudo e com os limites metodológicos assumidos, garantindo coerência entre o enquadramento teórico, a metodologia adotada e a natureza dos dados recolhidos.

1.4 Organização do documento

O Capítulo 1 apresenta o enquadramento geral da investigação, abordando de forma clara o problema de pesquisa, a motivação que impulsionou o estudo, os objetivos a serem alcançados e as questões que guiarão o desenvolvimento do trabalho.

No Capítulo 2, desenvolve-se a revisão da literatura, com uma análise aprofundada sobre o ensino e a aprendizagem de algoritmos, explorando as abordagens visuais e interativas e os

seus impactos. Além disso, são discutidos os princípios fundamentais da DSR, fornecendo o contexto teórico e metodológico para a pesquisa.

O Capítulo 3 descreve a metodologia adotada, oferecendo uma justificação detalhada para a escolha do modelo DSR-E, e explicando as etapas de aplicação desse modelo ao longo do projeto. Este capítulo foca em como o modelo orientou as decisões metodológicas e estruturou o desenvolvimento do artefacto.

No Capítulo 4, é abordado o processo de concepção, *design* e implementação do *Algoriddle*, com ênfase nos requisitos técnicos e funcionais que orientaram o desenvolvimento. A arquitetura do sistema e os algoritmos integrados são descritos em detalhe, explicando as escolhas e soluções adotadas para garantir a eficácia do ambiente interativo.

O Capítulo 5 analisa a avaliação do artefacto, descrevendo os métodos utilizados para a recolha e análise de dados. Este capítulo discute ainda os resultados obtidos, oferecendo uma análise crítica sobre a eficácia do *Algoriddle* e a perceção dos utilizadores, destacando tanto os pontos fortes como as áreas de melhoria.

Finalmente, o Capítulo 6 sintetiza as conclusões da investigação, refletindo sobre as limitações do estudo e apresentando recomendações para trabalhos futuros, com sugestões para a melhoria contínua do *Algoriddle* e a expansão da pesquisa a novos contextos.

O documento inclui ainda as referências bibliográficas e diversos apêndices que complementam e enriquecem a informação apresentada ao longo do trabalho.

2 Enquadramento Teórico

Este capítulo tem como objetivo fundamentar teoricamente a investigação, apresentando as bases conceptuais que sustentam o desenvolvimento do artefacto *Algoriddle*. Para tal, começa-se por abordar o ensino de algoritmos e os desafios a ele associados, seguido pela análise do papel das visualizações e da interatividade no processo de aprendizagem. Em seguida, discute-se as principais ferramentas disponíveis atualmente, destacando as suas limitações. Por fim, apresenta-se o enquadramento metodológico fornecido pela DSR.

2.1 O ensino de algoritmos e os seus desafios

Como foi referido no Capítulo 1, o ensino de algoritmos é um elemento central na formação em Ciências da Computação e Engenharia Informática, uma vez que constitui a base do desenvolvimento do pensamento computacional. Para (Wing, 2006), este pensamento envolve capacidades como a abstração, a decomposição e a formulação de procedimentos, sendo fundamental para compreender e resolver problemas computacionais. (Grover & Pea, 2013) reforçam esta ideia ao reconhecerem que a aprendizagem inicial de algoritmos envolve um conjunto complexo de processos cognitivos que os estudantes nem sempre dominam plenamente nas fases iniciais da sua formação.

As abordagens pedagógicas tradicionais apoiam-se frequentemente em explicações teóricas, pseudocódigo e exemplos estáticos. Contudo (Hundhausen *et al.*, 2002) assinalam que estas metodologias podem não tornar suficientemente visível a transformação das estruturas de dados ao longo da execução, dificultando a criação de modelos mentais consistentes. (Byrne *et al.*, 1999) referem ainda que a ausência de *feedback* dinâmico limita a capacidade dos estudantes para perceberem a relação entre as instruções do algoritmo e o comportamento resultante.

As perceções recolhidas junto de estudantes e profissionais de Informática, que serão analisadas com mais detalhe no Capítulo 5, fornecem evidências adicionais sobre as dificuldades e perceções associadas ao ensino e aprendizagem de algoritmos. Os profissionais entrevistados reconheceram que certos aspetos do raciocínio algorítmico podem continuar a apresentar desafios ao longo da prática profissional. Esta convergência entre os resultados encontrados na literatura e as perceções empíricas obtidas reforça a relevância do problema abordado neste estudo.

Assim, a necessidade de métodos que tornem mais clara a dinâmica interna dos algoritmos permanece atual e justifica a procura de estratégias de ensino assentes na visualização e na interatividade.

2.2 Visualização e interatividade no ensino

A visualização de algoritmos constitui uma estratégia amplamente explorada para apoiar a aprendizagem de processos computacionais. (Hundhausen *et al.*, 2002) demonstram que visualizar a evolução das estruturas de dados contribui para uma melhor compreensão dos passos executados por um algoritmo. Por seu lado (Ben-Bassat Levy *et al.*, 2003) ao estudarem o sistema *Jeliot* 2000, mostram como a representação gráfica de operações elementares facilita a ligação entre código, lógica e comportamento observável.

A literatura sugere que a eficácia da visualização aumenta quando combinada com interatividade. A possibilidade de alterar parâmetros, controlar a execução ou manipular cenários aproxima a aprendizagem do tipo de experimentação ativa descrita por (Piaget, 1970), que defende que o conhecimento se constrói a partir da ação. De modo semelhante, (Papert, 1980) argumenta que programar constitui uma forma de pensamento reflexivo, proporcionando um ambiente onde o aprendiz experimenta, erra, corrige e reformula. Embora estas bases teóricas não constituam evidência empírica direta sobre ferramentas visuais para algoritmos, fornecem um enquadramento conceptual relevante para compreender a importância da exploração ativa no domínio computacional.

(Naps *et al.*, 2002) sublinham que o envolvimento ativo do estudante na atividade é crucial para que a visualização tenha um impacto positivo na aprendizagem. Assim, não basta simplesmente assistir a animações, é necessário que o estudante manipule, explore e interaja de forma significativa com os conteúdos. De facto, estas conclusões alinham-se com as perceções dos inquiridos nesta investigação, os quais expressaram uma clara preferência por ferramentas que combinem uma visualização clara com um elevado grau de controlo sobre a execução.

2.2.1 Casos de sucesso noutras áreas

O uso de visualização dinâmica e interatividade não se limita ao ensino de algoritmos. Em diversas áreas científicas, estas abordagens demonstraram potencial para tornar acessíveis conteúdos abstratos.

Na matemática, o *GeoGebra* permite representar e manipular objetos e relações de forma dinâmica, favorecendo a compreensão de conceitos complexos (Hohenwarter & Gerber, 2009).

No ensino das ciências, as simulações do projeto *PhET Interactive Simulations* tornam possível explorar fenómenos físicos e químicos num ambiente controlado e visualmente apelativo, promovendo a aprendizagem baseada na descoberta (Wieman et al., 2012).

Em contextos de modelação científica, o *Modellus* permite articular representações matemáticas e simulações computacionais, evidenciando o potencial de ambientes que unem teoria, visualização e manipulação (Teodoro, 2002).

Estes exemplos reforçam a noção de que a representação dinâmica de processos complexos é uma estratégia transversal, potencialmente útil para o ensino de algoritmos.

2.3 Ferramentas visuais para o ensino de algoritmos

A aprendizagem de algoritmos tem beneficiado do desenvolvimento de ferramentas digitais que recorrem à visualização, animação e interatividade para apoiar a compreensão de conceitos estruturantes. Estas ferramentas procuram tornar explícitas operações que, quando apresentadas apenas em pseudocódigo ou texto, podem ser difíceis de interpretar, sobretudo para aprendentes em fases iniciais (Naps et al., 2002; Hundhausen et al., 2002).

Entre as diversas plataformas disponíveis *online* para a visualização de algoritmos e estruturas de dados encontram-se o *Algorithm Visualizer*, o *VisuAlgo*, o *Data Structure Visualizations*, o *BTree-js* e o *CS Unplugged*. Adicionalmente, recursos como o *BigO Cheatsheet* e o *Algorithms-DataStructures-BigONotation* apresentam sínteses visuais das complexidades temporais e espaciais de algoritmos

O *Algorithm Visualizer* (Figura 1) disponibiliza representações passo a passo de algoritmos clássicos, permitindo observar a evolução das estruturas de dados e o fluxo de execução. Esta plataforma recorre a animações sincronizadas e integra exemplos executáveis, oferecendo ao utilizador controlo sobre a velocidade e o avanço da execução.

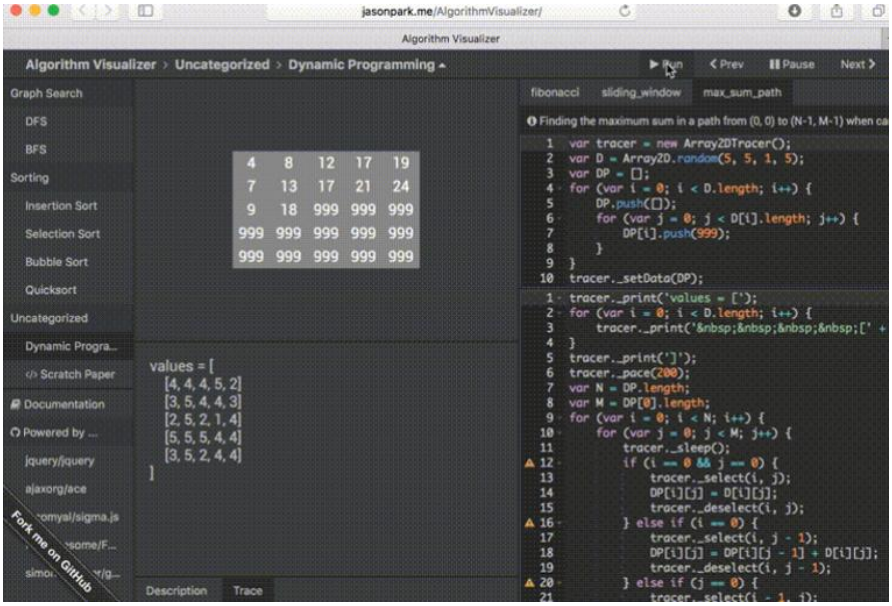


Figura 1 - *Algorithm Visualizer*

[Retirada de: https://light-city.medium.com/data-structure-mastery-unleashed-8-dynamic-online-visualization-platforms-fdc62bcc1225]

O *VisuAlgo* (Figura 2), desenvolvido na Universidade Nacional de Singapura, apresenta uma coleção extensiva de algoritmos e estruturas de dados. A sua abordagem pedagógica visa mostrar simultaneamente a lógica conceptual e as alterações no estado interno das estruturas, o que facilita a compreensão de relações entre operações.

Figura 2 - *VisuAlgo*

[Retirada de: https://light-city.medium.com/data-structure-mastery-unleashed-8-dynamic-online-visualization-platforms-fdc62bcc1225]

O *Data Structure Visualizations* (Figura 3), desenvolvido pela Universidade de São Francisco, incide na demonstração visual de operações fundamentais como inserções, remoções e reorganização de estruturas.

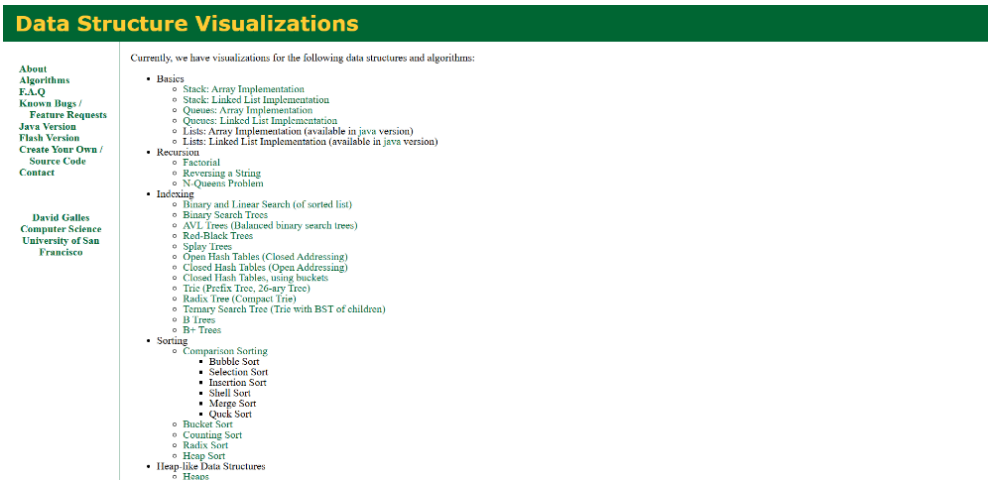


Figura 3 - *Data Structure Visualizations*

[Retirada de: https://light-city.medium.com/data-structure-mastery-unleashed-8-dynamic-online-visualization-platforms-fdc62bcc1225]

O *BTree-js* (Figura 4) centra-se exclusivamente na visualização de árvores B, permitindo observar processos de inserção, remoção, divisão de nós e reorganização hierárquica. O foco especializado torna-o particularmente útil no estudo de armazenamento e indexação de dados.

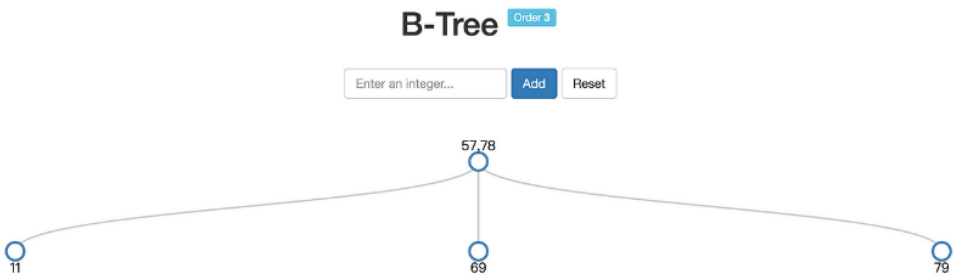


Figura 4 - *Btree-js*

[Retirada de: https://light-city.medium.com/data-structure-mastery-unleashed-8-dynamic-online-visualization-platforms-fdc62bcc1225]

O *CS Unplugged* (Figura 5) constitui uma abordagem distinta, ao propor atividades práticas e analógicas que ilustram conceitos algorítmicos sem recurso a computador. Trata-se de um recurso amplamente utilizado em níveis introdutórios e em ambientes com restrições tecnológicas.

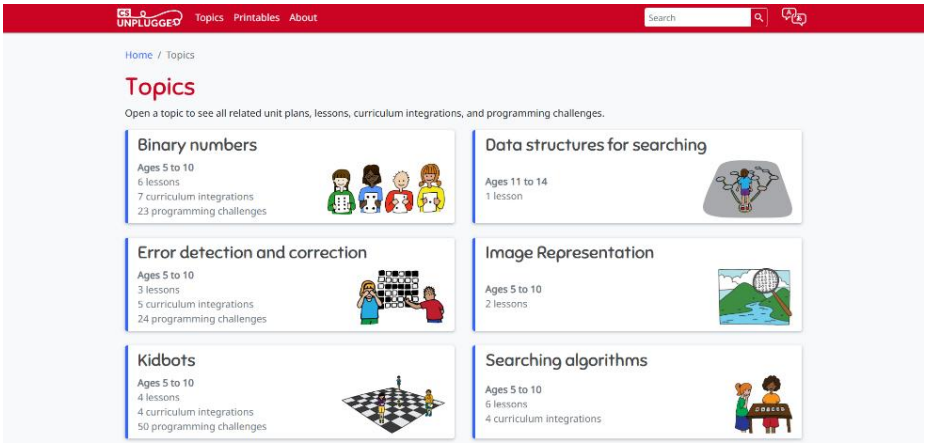
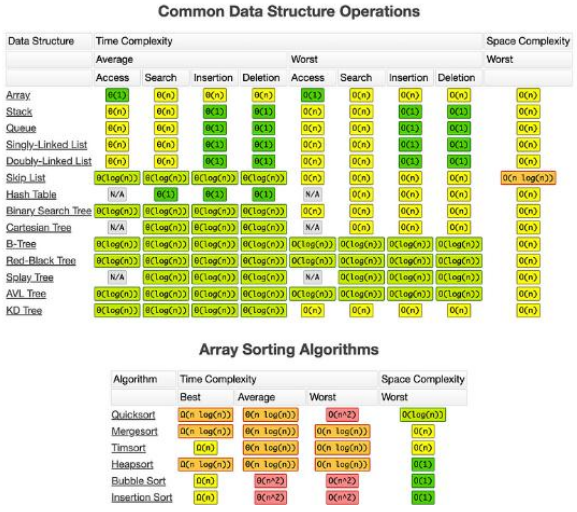


Figura 5 - *CS Unplugged*

[Retirada de: https://light-city.medium.com/data-structure-mastery-unleashed-8-dynamic-online-visualization-platforms-fdc62bcc1225]

Por fim, ferramentas como o *BigO Cheatsheet* (Figura 6) e o *Algorithms-DataStructures-BigONotation* (Figura 7) apresentam gráficos e tabelas que sintetizam a complexidade temporal e espacial de algoritmos e operações, funcionando como apoio conceptual na análise de eficiência.



Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)

Figura 6 - BigO Cheatsheet

[Retirada de: https://light-city.medium.com/data-structure-mastery-unleashed-8-dynamic-online-visualization-platforms-fdc62bcc1225]

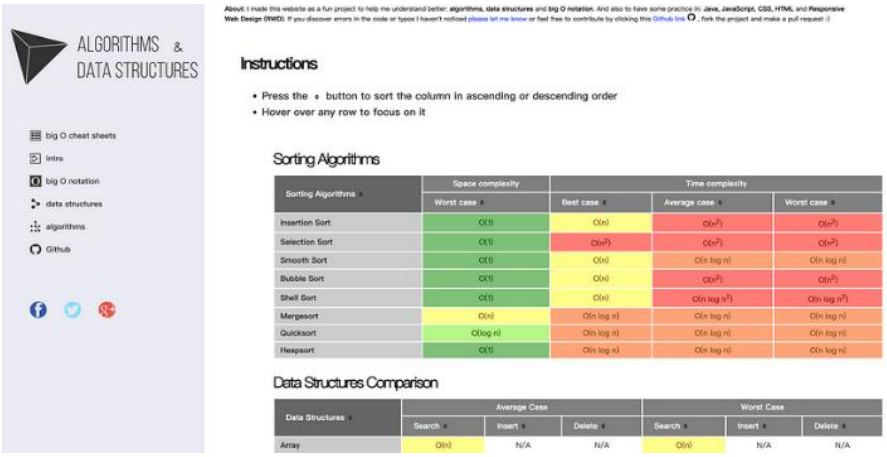


Figura 7 - Algorithms-DataStructures-BigONotation

[Retirada de: https://light-city.medium.com/data-structure-mastery-unleashed-8-dynamic-online-visualization-platforms-fdc62bcc1225]

Estas ferramentas, no seu conjunto, procuram tornar mais tangível a dinâmica de estruturas e algoritmos que, na sua expressão abstrata, podem ser de difícil assimilação.

2.3.1 Limitações das ferramentas analisadas

Apesar da diversidade de recursos disponíveis, a análise crítica das ferramentas, evidencia limitações que justificam o desenvolvimento de soluções alternativas.

Uma primeira limitação prende-se com o nível de interatividade. Entre as ferramentas analisadas, apenas o *Algorithm Visualizer* permite execução de código, embora com constrangimentos significativos ao nível da linguagem suportada e da complexidade dos exemplos. As restantes recorrem maioritariamente a simulações pré-definidas, o que reduz o envolvimento ativo do utilizador e limita a exploração de cenários personalizados.

A usabilidade e o *design* constituem outra limitação frequentemente referida pelos participantes. Em virtude disso, *interfaces* desatualizadas, a ausência de responsividade e a impossibilidade de personalização dificultam a integração em práticas pedagógicas contemporâneas, que valorizam acessibilidade e adaptação a diferentes perfis de utilizadores.

A limitada integração entre visualização e experimentação prática é igualmente relevante. Embora as animações tornem visível a dinâmica dos algoritmos, a maioria das ferramentas não permite ao utilizador testar variações, modificar parâmetros ou executar código próprio, reduzindo o potencial de aprendizagem ativa, aspeto amplamente reconhecido pela literatura como determinante para a eficácia das visualizações (Naps et al., 2002; Hundhausen et al., 2002).

Importa sublinhar que estas limitações são consistentes com críticas anteriores presentes na literatura. (Shaffer et al., 2007), num levantamento abrangente sobre visualizações de algoritmos, observaram que muitas ferramentas disponíveis careciam de interatividade significativa, apresentavam fraca integração pedagógica ou forneciam apoio limitado à exploração autónoma dos estudantes. Embora os autores não analisem especificamente as plataformas aqui descritas, as tendências identificadas mantêm-se pertinentes.

A Tabela 1 sintetiza comparativamente os principais aspetos identificados na análise das ferramentas estudadas.

Tabela 1 - Comparação entre ferramentas visuais e interativas para o ensino de algoritmos

	Interatividade	**Execução de código**	**Usabilidade / *Design***	**Suporte de múltiplas linguagens**
Algorithm Visualizer	**Moderado** (permite controlo passo a passo da execução)	Parcial (Python)	*Interface* intuitiva, mas limitada a versões *web*	Limitado
VisuAlgo	**Elevado** (inclui animações, pseudocódigo e explicações)	Não (simulação pré-definida)	Boa clareza visual, mas *design* estático	Não
Data Structure Visualizations	**Baixo** (apenas demonstrações visuais)	Não	Interface simples e pouco interativa	Não
Btree-js	**Médio** (manipulação direta da árvore B)	Não	Focado em aspetos técnicos, *interface* mínima	Não
CS Unplugged	**Alto** (atividades práticas offline)	Não aplicável	Excelente dinamismo, mas ausência de execução real	Não aplicável
BigO Cheatsheet / Algorithms-DataStructures-BigONotation	**Muito baixo** (apenas visualização comparativa)	Não	*Interface* informativa e estática	Não

2.4 *Design Science Research*

A DSR constitui uma metodologia orientada para a criação e avaliação de artefactos concebidos para resolver problemas reais. O enquadramento teórico da DSR tem origem na distinção proposta por (Simon, 1981) que distingue as ciências do artificial das ciências descritivas, ou seja, enquanto estas últimas procuram explicar fenómenos existentes, as primeiras centram-se no *design* de soluções para problemas concretos.

(Hevner et al., 2004) consolidam esta perspetiva ao propor um modelo composto por três ciclos fundamentais: o ciclo da relevância, o ciclo do rigor e o ciclo do *design*. Este modelo sublinha que a construção de artefactos úteis depende simultaneamente da identificação clara do problema, da fundamentação teórica sólida e da criação iterativa com validação constante.

(Peffers et al., 2007) complementam esta visão ao apresentar um processo metodológico estruturado para implementar a DSR, incluindo etapas que vão da identificação do problema à comunicação dos resultados.

2.4.1 Modelo de Hevner

O modelo de (Hevner et al., 2004) estrutura a investigação em três ciclos interdependentes anteriormente mencionados. Estes ciclos asseguram que a pesquisa não só resolve um problema real, mas também faz isso de maneira cientificamente fundamentada e através de um processo iterativo de *design*, conforme ilustrado na Figura 8.

O ciclo da relevância garante que o artefacto desenvolvido esteja diretamente alinhado com um problema real e significativo, sendo que a definição clara do problema é fundamental para assegurar que a solução proposta seja não apenas relevante, mas também aplicável ao contexto específico. Para isso, este ciclo envolve uma análise aprofundada das necessidades do público-alvo, de modo a garantir que o artefacto seja eficaz e útil na resolução do problema identificado.

O ciclo do rigor envolve o uso de teorias e métodos científicos consolidados para garantir que a solução criada tenha uma base sólida e seja validada cientificamente, distinguindo-se assim de metodologias puramente empíricas, ao incorporar uma fundamentação teórica robusta para justificar e sustentar as soluções propostas. Este ciclo assegura que o artefacto desenvolvido seja cientificamente válido e que a sua criação contribua para o avanço do conhecimento.

Por fim, o ciclo do *design* corresponde à fase de criação iterativa do artefacto, na qual são realizados constantes ajustes com base no *feedback* contínuo, fazendo com que o *design* passe por várias versões, que são avaliadas e melhoradas ao longo do processo. Esta abordagem

iterativa permite que a solução seja continuamente aprimorada, tornando-se mais eficaz e eficiente à medida que o desenvolvimento avança.

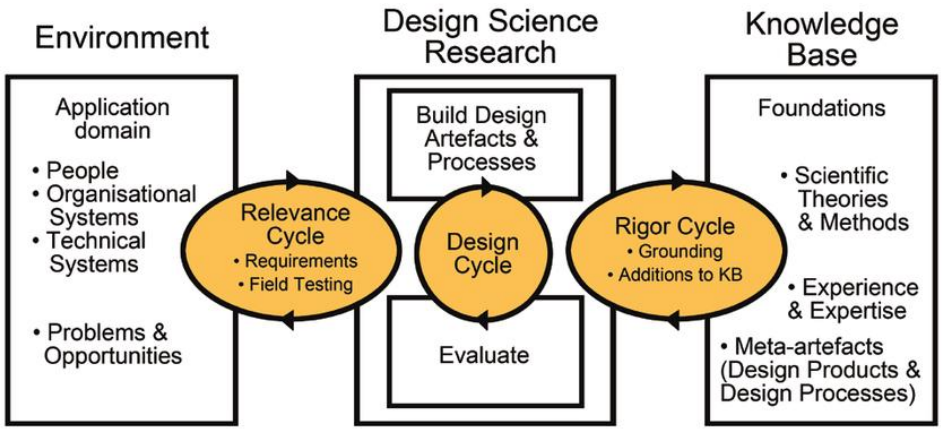


Figura 8 - Modelo dos três ciclos de Hevner

[Retirada de: https://www.researchgate.net/figure/Hevners-Design-Science-Research-Cycles-Hevner-et-al-2004-Hevner-2007_fig1_283417298]

Além dos ciclos, o modelo de (Hevner et al., 2004) define princípios que orientam o desenvolvimento e a avaliação de artefactos na DSR, garantindo que estes sejam úteis na prática e cientificamente rigorosos.

O desenvolvimento do artefacto deve gerar soluções viáveis, funcionais e relevantes para o contexto em que será aplicado. A avaliação deve ser sistemática e baseada em evidências, assegurando que a eficácia e a qualidade possam ser verificadas objetivamente.

A fundamentação teórica é essencial, pois fornece a base científica que sustenta as decisões de *design*. Ao mesmo tempo, o artefacto deve ter relevância prática, respondendo a necessidades reais e maximizando o impacto no contexto de aplicação.

O modelo também enfatiza iteratividade e *feedback* contínuo, permitindo melhorias constantes durante o desenvolvimento. A contribuição para o avanço científico é incentivada, garantindo que o artefacto gere novos conhecimentos ou *insights* significativos.

Por fim, a validação por pares assegura que a pesquisa e os artefactos sejam avaliados por especialistas, conferindo credibilidade, rigor e confiabilidade aos resultados.

De forma resumida, esses princípios orientam todo o processo de criação e avaliação, garantindo que os artefactos sejam inovadores, eficazes e cientificamente fundamentados.

2.4.2 Modelo DSR Educacional

Com base nos princípios da DSR, (Pimentel et al., 2020) propuseram o modelo DSR-E, que adapta a metodologia DSR às especificidades da investigação em educação e tecnologias educativas. Este modelo considera o artefacto educacional, seja um ambiente digital, recurso pedagógico ou metodologia, como sendo um objeto e instrumento de investigação, permitindo tanto a experimentação prática quanto a geração de conhecimento científico.

O DSR-E organiza-se em quatro dimensões principais. A primeira é a definição do problema em contexto, que identifica claramente a questão educacional e o público-alvo, assegurando que o artefacto responda a necessidades reais. A segunda dimensão centra-se no artefacto, abrangendo a sua concepção e desenvolvimento como solução concreta para o problema identificado. A terceira dimensão corresponde às conjecturas comportamentais, ou seja, hipóteses sobre como o artefacto poderá influenciar a aprendizagem e a motivação dos estudantes. Por fim, a quarta dimensão refere-se à avaliação empírica, que envolve a recolha e análise sistemática de evidências para validar ou ajustar as conjecturas, promovendo um ciclo contínuo de melhoria do artefacto. Estas quatro dimensões são representadas na Figura 9, evidenciando a interação iterativa e reflexiva entre elas.

A dinâmica entre estas dimensões favorece um processo iterativo e reflexivo, em que cada ciclo de *design* contribui simultaneamente para aperfeiçoar o artefacto e para o avanço do conhecimento científico sobre ensino-aprendizagem. O investigador atua como designer reflexivo, colaborando ativamente com os estudantes e ajustando o artefacto com base no feedback obtido durante a implementação.

No âmbito desta dissertação, o modelo DSR-E foi adotado por oferecer uma estrutura capaz de conciliar rigor científico e relevância pedagógica, permitindo compreender de que forma a interação visual e a experiência prática podem melhorar a aprendizagem de algoritmos, indo além da criação de uma simples ferramenta tecnológica.

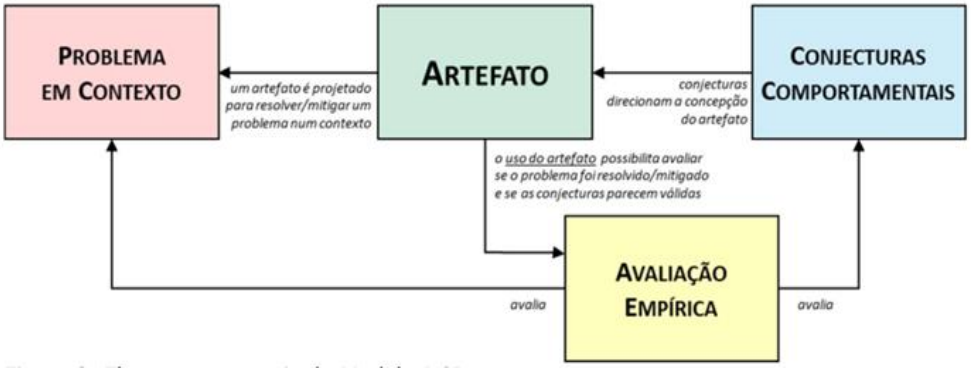


Figura 9 - Modelo DSR Educacional

[Retirada de: (Pimentel et al., 2020)]

2.4.3 Justificação da escolha do modelo

A escolha do modelo DSR-E de (Pimentel et al., 2020) justifica-se pela sua capacidade de conciliar inovação tecnológica com investigação aplicada em contextos educativos. Diferentemente de metodologias como o *Constructive Approach* (CA) de (Kasanen et al., 1991), cuja validação teórica ocorre *ex post* e sem ciclos iterativos de avaliação, o DSR-E permite que o desenvolvimento do artefacto e a análise do seu impacto pedagógico ocorram em paralelo.

O modelo reconhece o artefacto educacional como simultaneamente objeto e instrumento de investigação, permitindo que cada iteração seja guiada por dados empíricos sobre a interação dos participantes, as suas dificuldades, estratégias de aprendizagem e perceções sobre a usabilidade e relevância pedagógica. Essa abordagem iterativa e reflexiva assegura que o artefacto seja constantemente ajustado às necessidades reais dos aprendentes, aumentando a eficácia do processo de ensino-aprendizagem.

Além disso, o DSR-E integra de forma sistemática a formulação de conjecturas comportamentais, o desenvolvimento de soluções concretas e a avaliação empírica, promovendo ciclos contínuos de melhoria fundamentados em evidências observáveis. Essa estrutura permite não apenas a criação de uma ferramenta tecnológica funcional, mas também a geração de conhecimento científico sobre como a visualização e a interatividade contribuem para a aprendizagem de algoritmos.

Portanto, a opção pelo DSR-E garante que o desenvolvimento do *Algoriddle* seja simultaneamente inovador, fundamentado e relevante, permitindo uma investigação aplicada robusta, que incorpora perceções empíricas dos estudantes e consistentes com os objetivos da investigação educativa.

3 Metodologia

O desenvolvimento e a avaliação do *Algoriddle* seguiram o modelo DSR-E, apresentado no Capítulo 2, que orienta a criação de artefactos educacionais através de ciclos iterativos que articulam claramente o problema identificado, a fundamentação teórica, o *design* do artefacto e a validação empírica dos resultados. Este modelo permite que as decisões metodológicas mantenham alinhamento contínuo com os objetivos e questões de investigação formulados no Capítulo 1.

No contexto desta dissertação, o DSR-E foi aplicado em seis fases: definição do problema, revisão da literatura, sugestão de soluções, desenvolvimento do artefacto, avaliação e comunicação dos resultados. As secções seguintes descrevem o modo como cada fase foi operacionalizada, conforme ilustrado na Figura 10.

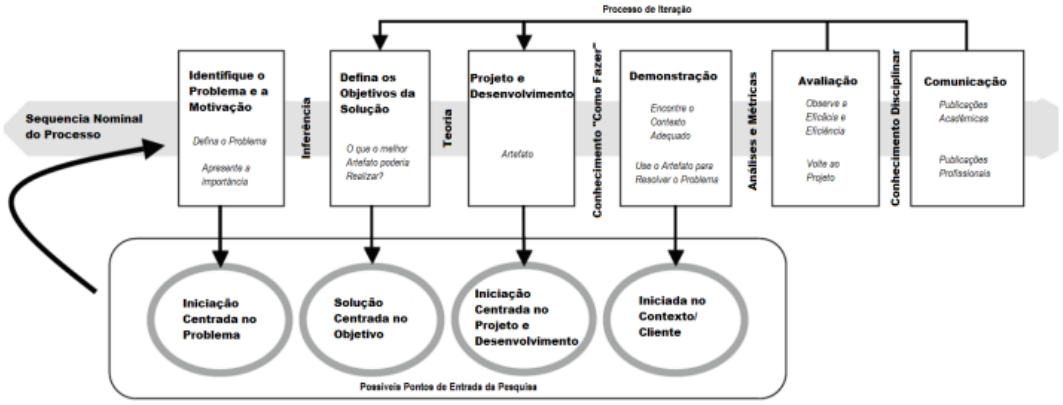


Figura 10 – Fases do modelo DSR-E (adaptado do diagrama do processo de metodologia DSR)

[Retirada de: (Peffers et al., 2007)]

3.1 Fases do modelo DSR Educacional

3.1.1 Definição do problema

A investigação iniciou-se com a identificação clara do problema educacional que motivou o estudo. Tal como discutido no Capítulo 1 e sustentado por (Byrne et al., 1999), (Hundhausen et al., 2002) e (Shaffer et al., 2007), os métodos tradicionais de ensino de algoritmos, baseados essencialmente em pseudocódigo, explicações teóricas e exemplos estáticos, tendem a

promover uma compreensão superficial dos conceitos fundamentais. Esta limitação torna-se ainda mais evidente na ausência de *feedback* imediato e na reduzida oferta de oportunidades para experimentação ativa, aspetos considerados essenciais para o desenvolvimento do pensamento computacional (Wing, 2006; Grover & Pea, 2013).

A partir deste diagnóstico, verificou-se a necessidade de explorar soluções tecnológicas capazes de promover visualização, interatividade e execução prática de código, de forma a apoiar a construção de modelos mentais mais sólidos por parte dos estudantes de Engenharia Informática.

3.1.2 Revisão da literatura

A revisão da literatura corresponde ao estado da arte apresentado no Capítulo 2 e fornece o enquadramento teórico que sustenta esta investigação. A literatura demonstra que abordagens baseadas em visualização e interatividade podem apoiar significativamente a compreensão de algoritmos, uma vez que promovem envolvimento ativo e permitem observar a dinâmica dos processos algorítmicos (Hundhausen et al., 2002; Ben-Bassat Levy et al., 2003).

Em paralelo, perspetivas construtivistas e construcionistas, associadas aos trabalhos de (Piaget, 1970) e (Papert, 1980), reforçam a importância de ambientes que favoreçam experimentação, manipulação e construção ativa do conhecimento.

Ao nível metodológico, o modelo DSR-E, descrito por (Pimentel et al., 2020) e alinhado com os princípios estruturantes de (Hevner et al., 2004), oferece a base conceptual necessária para orientar o desenvolvimento, validação e comunicação de artefactos tecnológicos. Este conjunto de contributos sustentou a identificação das lacunas existentes nas soluções atuais e definiu os requisitos orientadores das etapas subsequentes.

3.1.3 Sugestão de soluções

Com base nas lacunas identificadas, foi proposta a criação do *Algoriddle*, um ambiente digital que integra visualização animada de algoritmos, manipulação de parâmetros, execução controlada de código e mecanismos de *feedback* imediato. Estas funcionalidades foram conceptualizadas de forma a responder às necessidades evidenciadas na literatura e às dificuldades relatadas pelos estudantes desta pesquisa no processo de aprendizagem de algoritmos.

3.1.4 Desenvolvimento do artefacto

O desenvolvimento do *Algoriddle* foi conduzido de forma iterativa, seguindo ciclos contínuos de prototipagem, teste e refinamento, conforme preconizado pelo modelo DSR-E. O Capítulo 4 descreve detalhadamente este processo, abrangendo a definição de requisitos, a arquitetura do sistema, as funcionalidades implementadas e a respetiva fundamentação técnica e pedagógica.

Adicionalmente, o Capítulo 4 apresenta um levantamento dos planos de estudos das principais universidades e institutos politécnicos públicos de Portugal, realizado com o propósito de identificar os algoritmos mais frequentemente lecionados no ensino superior. Esta análise permitiu selecionar algoritmos alinhados com a prática curricular nacional, reforçando a pertinência e aplicabilidade do *Algoriddle* no contexto educativo português.

Cada iteração do processo de desenvolvimento teve como objetivo assegurar que o artefacto refletia de forma adequada os requisitos funcionais e não funcionais identificados, bem como as necessidades dos utilizadores finais. Este ciclo de aperfeiçoamento progressivo contribuiu para a construção de um sistema coerente, relevante e pedagogicamente fundamentado.

Embora o modelo DSR-E tenha orientado o processo global de investigação, o desenvolvimento do *Algoriddle* incorporou práticas comuns a metodologias iterativas da área de Engenharia de *Software*, nomeadamente ciclos curtos de implementação, teste e refinamento funcional. Estas práticas, inspiradas em abordagens ágeis, permitiram ajustar progressivamente a *interface*, as funcionalidades e os mecanismos de interação com base na experimentação contínua do artefacto, ainda que não tenha sido seguida formalmente uma metodologia ágil específica como *Scrum* ou XP.

3.1.5 Avaliação do artefacto

Após o desenvolvimento, foi conduzida uma avaliação sistemática para analisar a usabilidade, clareza e perceções dos utilizadores relativamente ao *Algoriddle*. A avaliação incidiu essencialmente sobre estudantes do ensino superior na área da informática, conforme descrito no Capítulo 5, permitindo recolher evidências sobre a experiência de utilização e identificar oportunidades de melhoria.

Embora o foco formal da avaliação recaísse sobre estudantes, foram igualmente recolhidas perceções de profissionais da área da programação, que possuem experiência consolidada em desenvolvimento e análise de *software*. Estas opiniões, ainda que não integrem a avaliação

empírica principal, contribuíram para uma apreciação técnica complementar do artefacto e forneceram um enquadramento adicional relevante durante o processo de melhoria.

Importa salientar que não foram avaliados impactos pedagógicos em contexto de sala de aula, nem foram recolhidas perceções de docentes. Consequentemente, os resultados obtidos referem-se exclusivamente à interação dos participantes com o artefacto e à sua perceção subjetiva de utilidade e usabilidade. Estes aspetos são discutidos como limitações no Capítulo 6.

3.1.6 Comunicação dos resultados

A fase final corresponde à documentação e disseminação dos resultados. A presente dissertação constitui o principal instrumento de comunicação científica deste estudo, descrevendo todo o processo de concepção, desenvolvimento e avaliação do artefacto.

O objetivo desta etapa é tornar claro o contributo do *Algoriddle* para a área das tecnologias educativas e apoiar futuras investigações. Embora se preveja a elaboração de um artigo científico com base nos resultados obtidos, este trabalho não realiza reivindicações sobre impacto pedagógico direto, dado que tal não foi avaliado em contexto de sala de aula, sendo que, investigações futuras poderão aprofundar essa dimensão.

3.2 Métodos de recolha e análise de dados

A presente secção descreve os procedimentos adotados para a recolha e análise de dados que sustentam a avaliação do *Algoriddle*. Estes métodos foram definidos de forma a garantir rigor, coerência metodológica e alinhamento com os objetivos do estudo, proporcionando uma compreensão clara das etapas de recolha, dos participantes envolvidos e dos critérios utilizados na interpretação dos resultados.

3.2.1 Recolha de dados

A recolha de dados ocorreu em duas etapas distintas. A primeira etapa, de caráter exploratório e apresentada no Capítulo 2, consistiu na recolha de perceções gerais sobre desafios associados ao ensino de algoritmos, envolvendo estudantes e profissionais da área da Informática. Estas perceções, discutidas à luz de autores como (Hundhausen et al., 2002) e (Byrne et al., 1999), permitiram aprofundar a compreensão das dificuldades enfrentadas pelos

aprendentes. O objetivo desta etapa foi o de apoiar a identificação do problema e a definição dos requisitos iniciais.

A segunda etapa corresponde à avaliação formal do *Algoriddle*, descrita no Capítulo 5. Esta fase envolveu quinze participantes na área da Informática, que interagiram com o artefacto e responderam a instrumentos de recolha de dados de natureza quantitativa e qualitativa. Os instrumentos utilizados foram inspirados em (Brooke, 1996), bem como nas orientações metodológicas de (Hevner et al., 2004) e (Pimentel et al., 2020), permitindo analisar usabilidade, clareza e perceções de utilidade. Esta avaliação não teve como objetivo medir impacto pedagógico direto, uma vez que tal exigiria aplicação em contexto de sala de aula.

Esta distinção entre etapas assegura coerência metodológica e impede extrapolações indevidas sobre os resultados obtidos.

3.2.2 Considerações éticas

Em termos éticos, a investigação cumpriu integralmente os princípios da investigação científica e as normas estabelecidas pelo Regulamento Geral sobre a Proteção de Dados (RGPD). O estudo foi previamente submetido à comissão de ética institucional, que confirmou que, devido à natureza anónima dos instrumentos de recolha, não era necessário solicitar parecer ao Encarregado de Proteção de Dados (EPD). Caso fossem recolhidos dados pessoais identificáveis, teria sido exigida a emissão de um parecer formal, o que não se aplicou neste caso.

O questionário utilizado foi totalmente anónimo, não recolhendo quaisquer dados sensíveis ou identificadores pessoais. O único dado sociodemográfico recolhido foi o nível de experiência em programação, utilizado apenas para análise agregada e sem possibilidade de identificar individualmente os participantes. Todos os participantes foram plenamente informados sobre os objetivos da investigação e participaram de forma voluntária, tendo a possibilidade de desistir a qualquer momento sem qualquer prejuízo.

Os dados recolhidos foram tratados exclusivamente para fins académicos, apresentados de forma agregada e anónima, assegurando a confidencialidade, privacidade e integridade dos participantes. Esta atenção às questões éticas reforça a natureza responsável da investigação e garante conformidade com os princípios de proteção de dados aplicáveis à investigação científica. Desta forma, a metodologia adotada permitiu conduzir uma avaliação rigorosa e integrada do *Algoriddle*, combinando evidências objetivas sobre usabilidade e clareza com perceções subjetivas dos utilizadores, sem comprometer a proteção dos participantes nem violar normas éticas ou legais.

4 Desenvolvimento

O presente capítulo descreve detalhadamente o processo de concepção, *design* e implementação do *Algoriddle*, artefacto educacional desenvolvido no âmbito desta investigação. A materialização deste ambiente virtual resulta da aplicação iterativa do modelo DSR-E, descrito no Capítulo 3, alinhando rigor tecnológico com objetivos educacionais concretos, sobretudo a necessidade de apoiar a compreensão de algoritmos por parte de estudantes do ensino superior em Informática.

A estrutura deste capítulo segue as etapas que compõem o ciclo de desenvolvimento do artefacto: definição dos objetivos e requisitos, análise curricular para fundamentar a seleção dos algoritmos, concepção arquitetónica, integração de princípios de usabilidade e inclusão e, finalmente, a modelação do sistema e da interação com o utilizador.

O propósito é demonstrar como o *design* do *Algoriddle* responde diretamente ao problema identificado na fase inicial da investigação e como cada decisão contribui para a criação de uma ferramenta interativa que potencia a observação e exploração de algoritmos em execução.

4.1 Concepção do artefacto

A concepção do *Algoriddle* partiu da constatação de que os métodos de ensino com forte dependência da exposição teórica e da apresentação estática de pseudocódigo frequentemente conduzem a uma compreensão limitada dos algoritmos, especialmente no que se refere ao encadeamento lógico das operações e às alterações dinâmicas das estruturas de dados durante a execução. A ausência de representações mais concretas dificulta que os estudantes consigam visualizar o que efetivamente acontece nos bastidores de um programa, tornando-se mais complexo desenvolver raciocínio algorítmico consistente.

Com base neste desafio educacional, definiu-se que a experiência de aprendizagem proporcionada pelo artefacto deveria permitir ao estudante acompanhar passo a passo a execução de um algoritmo, observando, em simultâneo, a animação gráfica, as alterações de variáveis e estruturas, e o código subjacente. O *Algoriddle* foi, assim, concebido como um ambiente digital que transformasse a aprendizagem teórica numa experiência exploratória e interativa, possibilitando uma melhor construção conceptual. A plataforma foi pensada para ser utilizada tanto em sala de aula como em contextos autónomos, envolvendo estudantes com diferentes níveis de experiência em programação .

Esta concepção seguiu uma abordagem iterativa baseada em ciclos de prototipagem, testes e ajustamentos progressivos. Em cada ciclo foram recolhidos contributos informais de utilizadores, que permitiram refinar aspetos da *interface* e da interação, garantindo que o desenvolvimento se mantivesse alinhado com as necessidades reais da aprendizagem.

4.2 Requisitos e objetivos de desenvolvimento

A definição dos requisitos que nortearam o desenvolvimento do *Algoriddle* resultou da necessidade de equilibrar duas dimensões complementares: a dimensão técnica, que assegura estabilidade, desempenho, fiabilidade e compatibilidade transversal do sistema, e a dimensão pedagógica, que garante condições adequadas para a promoção da compreensão algorítmica e da autonomia do estudante.

4.2.1 Enquadramento e fundamentação

Do ponto de vista pedagógico, tornou-se essencial garantir que o sistema oferecesse representações visuais animadas e sincronizadas com o pseudocódigo, bem como explicações adicionais em linguagem natural. Esta articulação entre o que o algoritmo faz e o que o estudante vê contribuiria para uma compreensão mais profunda e significativa dos conceitos. Do ponto de vista técnico, a aplicação deveria assegurar fluidez e responsividade em diferentes dispositivos, promover acessibilidade e manter uma estrutura modular que permitisse futuras expansões, quer ao nível das funcionalidades, quer ao nível dos algoritmos incluídos.

A aplicação destes princípios enquadrou-se nos pressupostos do modelo DSR-E, que defende a articulação entre rigor metodológico, relevância prática e melhoria iterativa do artefacto ao longo do processo de desenvolvimento.

4.2.2 Recolha e análise de dados

A recolha e análise de dados que sustentaram a definição dos requisitos do *Algoriddle* assentaram numa abordagem metodológica coerente com o modelo DSR-E, orientada para a identificação rigorosa das necessidades pedagógicas e tecnológicas associadas ao ensino de algoritmos. Esta etapa teve como referência os desafios descritos e os objetivos educacionais e funcionais delineados nos capítulos anteriores, garantindo a rastreabilidade entre a fundamentação teórica e as especificações do artefacto.

Os requisitos foram organizados em duas categorias, funcionais e não funcionais, assegurando simultaneamente a adequação pedagógica, técnica e de usabilidade da aplicação. Esta classificação permitiu estruturar o desenvolvimento de modo a garantir que o artefacto respondesse eficazmente às necessidades dos utilizadores e aos propósitos educativos que guiaram o *design* do sistema.

A cada requisito foi atribuída uma prioridade específica, distinguindo-se entre *Must Have*, funcionalidades essenciais sem as quais o sistema não cumpriria os objetivos propostos, *Should Have*, características desejáveis que enriquecem a experiência de aprendizagem e *Could Have*, elementos não críticos mas que contribuem para a evolução futura do artefacto e para uma utilização mais completa e motivadora. Esta distinção foi decisiva para orientar o planeamento iterativo e a priorização das tarefas de desenvolvimento ao longo das várias fases do projeto.

Todos os requisitos foram formulados com estrutura normativa explícita, iniciando-se com a expressão **"**O sistema deve…". Esta opção seguiu as recomendações do (IEEE, 2011) para a engenharia de requisitos, garantindo consistência descritiva, clareza comunicacional e verificabilidade objetiva das funcionalidades, fatores fundamentais para assegurar a rastreabilidade ao longo do ciclo de vida do sistema e a comunicação eficaz entre os diferentes intervenientes no processo de desenvolvimento.

4.2.3 Requisitos funcionais

Os requisitos funcionais do sistema definiram as suas capacidades essenciais, incluindo a disponibilização de animações sequenciais para a execução dos algoritmos, a execução de código em diferentes linguagens de programação, a interação com parâmetros de entrada, o controlo do fluxo de execução e a disponibilização de elementos informativos complementares que auxiliam na interpretação dos passos do algoritmo. Estes requisitos foram concebidos para garantir que o artefacto cumprisse o objetivo central de apoiar a compreensão conceptual dos estudantes, conforme sintetizado na Tabela 2.

A priorização destes requisitos seguiu a técnica *MoSCoW*, amplamente utilizada em Engenharia de *Software* e em contextos de desenvolvimento iterativo. Esta técnica classifica os requisitos em "*Must have*", "*Should have*" e "*Could have*", permitindo distinguir claramente as funcionalidades essenciais (*Must have*), aquelas que acrescentam valor significativo à experiência do utilizador (*Should have*), e funcionalidades complementares ou desejáveis, mas não críticas para o cumprimento dos objetivos centrais do artefacto (*Could have*). A adoção desta abordagem revelou-se eficaz no contexto do projeto, pois facilitou a tomada de decisões durante o desenvolvimento e assegurou que o foco estivesse nas funcionalidades mais importantes para o funcionamento do *Algoriddle*.

Tabela 2 - Requisitos funcionais

	Descrição	Prioridade	Justificação	Desenvolvido
RF01	O sistema deve executar código em ambiente seguro.	*Must Have*	Permitir ao utilizador testar algoritmos sem riscos de segurança.	Sim
RF02	O sistema deve disponibilizar um editor de código com realce de sintaxe.	*Must Have*	Facilita a leitura e escrita de código, melhorando a experiência do utilizador.	Sim
RF03	O sistema deve permitir a seleção de algoritmos a partir de uma lista predefinida.	*Must Have*	Facilita a escolha do algoritmo a estudar ou executar.	Sim
RF04	O sistema deve fornecer uma visualização passo a passo da execução do algoritmo.	*Must Have*	Essencial para a compreensão do funcionamento interno do algoritmo.	Sim
RF05	O sistema deve disponibilizar controlos de execução (iniciar, pausa, avançar e retroceder, reiniciar).	*Must Have*	Concede ao utilizador controlo total sobre o ritmo da visualização.	Sim
RF06	O sistema deve apresentar descrições e explicações teóricas sobre o algoritmo selecionado.	*Must Have*	Apoia a compreensão conceptual e reforça a vertente pedagógica.	Sim
RF07	O sistema deve exibir indicadores de complexidade temporal e espacial.	*Must Have*	Fornece informação essencial para a análise da eficiência algorítmica.	Sim
RF08	O sistema deve permitir a importação de dados.	*Should Have*	Viabiliza a experimentação de algoritmos com diferentes conjuntos de dados	Sim
RF09	O sistema deve permitir a exportação de resultados e código fonte	*Could Have*	Facilita a partilha e reutilização do trabalho do utilizador	Não
RF10	O sistema deve suportar multilingue.	*Must Have*	Garante a acessibilidade a utilizadores de diferentes nacionalidades.	Sim
RF11	O sistema deve integrar um *chatbot* de ajuda contextual.	*Should Have*	Melhora o suporte e esclarecimento de dúvidas durante a utilização.	Sim

RF12	O sistema deve incluir um motor de pesquisa de algoritmos.	*Should Have*	Facilita a localização rápida de algoritmos específicos.	Não
RF13	O sistema deve permitir a personalização do tema (modo escuro e claro).	Could Have	Melhora o conforto visual e a acessibilidade.	Sim
RF14	O sistema deve registar e disponibilizar o histórico de execuções anteriores.	*Could Have*	Permite rever execuções anteriores para fins de comparação.	Não
RF15	O sistema deve suportar múltiplas linguagens de programação (ex: *Python*, *JavaScript*, *C++, Java*)	*Could Have*	Aumenta a abrangência da aplicação e a sua utilidade pedagógica.	Sim
RF16	O sistema deve integrar com plataformas externas (ex: *GitHub*, *Google Drive*)	*Could Have*	Facilita a importação e exportação de código e dados.	Não
RF17	O sistema deve exibir feedback imediato em caso de erro de sintaxe ou execução.	*Must Have*	Ajuda o utilizador a identificar e corrigir erros rapidamente.	Sim
RF18	O sistema deve permitir editar e guardar algoritmos personalizados.	*Should Have*	Incentiva a criatividade e a personalização do conteúdo.	Não
RF19	O sistema deve apresentar visualizações gráficas (ex: árvores, grafos, *arrays*).	*Must Have*	Essencial para a compreensão visual de estruturas de dados e algoritmos.	Sim
RF20	O sistema deve incluir uma secção de *FAQs* e documentação de apoio.	*Should Have*	Ajuda na resolução de dúvidas frequentes e promove a autonomia.	Sim
RF21	O sistema deve garantir a acessibilidade através de atalhos, navegação por teclado e leitores de ecrã.	*Should Have*	Assegura a inclusão de utilizadores com necessidades especiais.	Não

4.2.4 Requisitos não funcionais

A definição dos requisitos não funcionais assegurou a qualidade global do artefacto, contemplando aspetos essenciais como usabilidade, desempenho, portabilidade, manutenibilidade e segurança. Entre outros, definiu-se que o sistema deveria possuir uma *interface* intuitiva, responder rapidamente às ações do utilizador, ser compatível com os

principais navegadores, adotar uma arquitetura modular de fácil manutenção, executar código do utilizador em ambiente seguro (*sandbox*) e proteger dados pessoais sensíveis.

Como apresentado na Tabela 3, estes requisitos estabeleceram os parâmetros de qualidade necessários para garantir o funcionamento consistente, seguro e fiável do artefacto.

Tabela 3 - Requisitos não funcionais

	Descrição	**Prioridade**	**Tipo**	**Justificação**
RNF01	O sistema deve possuir uma *interface* intuitiva e de fácil utilização.	*Must Have*	Usabilidade	Reduz a curva de aprendizagem e aumenta a satisfação do utilizador.
RNF02	O sistema deve responder em menos de 2 segundos em execuções simples.	*Must Have*	Performance	Garante fluidez e evita frustração do utilizador.
RNF03	O sistema deve ser compatível com os principais navegadores (*Chrome, Firefox, Edge, Safari*).	*Must Have*	Portabilidade	Aumenta o alcance e acessibilidade da aplicação.
RNF04	O sistema deve suportar dispositivos móveis e *tablets*.	*Should Have*	Portabilidade	Permite o uso da plataforma em qualquer contexto ou dispositivo.
RNF05	O sistema deve adotar uma arquitetura modular e de fácil manutenção.	*Must Have*	Manutenibilidade	Facilita a correção de erros e adição de novas funcionalidades.
RNF06	O sistema deve incluir documentação técnica e funcional clara e atualizada.	*Should Have*	Manutenibilidade	Apoia programadores e utilizadores na compreensão do sistema.
RNF07	O sistema deve executar código do utilizador em ambiente isolado (*sandbox*).	*Must Have*	Segurança	Impede ataques e protege o servidor.
RNF08	O sistema deve garantir a proteção de dados pessoais e sensíveis.	*Must Have*	Segurança	Cumpre os requisitos legais e o RGPD.
RNF09	O sistema deve facilitar a adição de novos idiomas sem necessidade de reestruturação.	*Should Have*	Internacionalização	Permite expansão para novos públicos.

RNF10	O sistema deve cumprir as normas de acessibilidade *WCAG 2.1 AA*.	*Should Have*	Acessibilidade	Garante o acesso universal a utilizadores com deficiência.
RNF11	O sistema deve permitir atualizações sem perda de dados do utilizador.	*Should Have*	Fiabilidade	Evita perda de informação e frustração do utilizador.
RNF12	O sistema deve registar logs de erros e métricas de desempenho.	*Should Have*	Fiabilidade	Facilita a deteção e resolução de problemas.
RNF13	O sistema deve suportar múltiplos utilizadores em simultâneo.	*Could Have*	Escalabilidade	Viabiliza o uso em contexto educativo coletivo.
RNF14	O sistema deve efetuar *backups* automáticos dos dados do utilizador.	*Could Have*	Fiabilidade	Previne perda de dados em caso de falha.
RNF15	O sistema deve ter instalação e configuração simplificadas.	*Should Have*	Usabilidade	Reduz barreiras à adoção e implementação.
RNF16	O sistema deve otimizar o consumo de recursos (*CPU*, *RAM*)	*Should Have*	Performance	Assegura bom desempenho mesmo em dispositivos modestos.

4.3 Análise curricular e seleção de algoritmos

A seleção dos algoritmos a incluir no *Algoriddle* baseou-se numa análise curricular sistemática, orientada para assegurar que os conteúdos integrados na aplicação fossem coerentes com a prática pedagógica prevalecente no ensino superior português. Para tal, foi realizado um levantamento dos planos curriculares das principais universidades e institutos politécnicos, apresentado nas Tabelas 4 e 5, complementado por uma análise detalhada dos conteúdos efetivamente lecionados nas unidades curriculares de Algoritmos e Estruturas de Dados, ilustrada nas Tabelas 6 e 7 .

Esta análise permitiu identificar uma forte convergência temática ao nível dos algoritmos fundamentais para a formação inicial, nomeadamente os algoritmos de ordenação, de pesquisa e os percursos em grafos, os quais constituem pilares essenciais do desenvolvimento do raciocínio algorítmico nos primeiros anos dos cursos de Informática. Deste modo, tornou-se

possível garantir que o artefacto a desenvolver se articulasse de forma válida com o contexto curricular onde se pretende integrar.

A pertinência destes algoritmos foi ainda analisada à luz das diretrizes internacionais definidas pela *Association for Computing Machinery* (ACM) e pelo *Institute of Electrical and Electronics Engineers* (IEEE), que estabelecem os conteúdos basilares da área da Ciência de Computadores. De acordo com o documento ACM/IEEE *Computer Science Curricula 2023*, a compreensão da complexidade de algoritmos, o domínio de estruturas de dados lineares e hierárquicas, os algoritmos de ordenação e pesquisa e os algoritmos em grafos constituem competências estruturantes da formação em Computação. A convergência entre estas recomendações e as práticas verificadas nas instituições portuguesas reforça a validade científica e educacional da seleção realizada, assegurando que o *Algoriddle* se encontra alinhado com exigências pedagógicas amplamente reconhecidas no ensino superior da área.

Assim, o processo de decisão sobre os algoritmos a implementar levou em consideração tanto a relevância dos conteúdos no panorama nacional como a sua consistência relativamente às orientações internacionais, garantindo que a plataforma se constitua como um recurso adequado ao contexto real de aprendizagem e alinhado com o que os estudantes necessitam de desenvolver nesta fase da sua formação académica.

Tabela 4 - Principais universidades públicas de Portugal

Universidade	**Localização**	**Plano curricular**
Instituto Superior Técnico (IST)	Lisboa	Sim
Instituto Universitário de Lisboa (ISCTE)	Lisboa	Sim
Universidade do Algarve (UAlg)	Faro	Sim
Universidade de Aveiro (UA)	Aveiro	Sim
Universidade dos Açores (UAc)	Ponta Delgada	Sim
Universidade da Beira Interior (UBI)	Covilhã	Sim
Universidade de Coimbra (UC)	Coimbra	Sim
Universidade de Évora (UEvora)	Évora	Sim
Universidade de Lisboa (ULisboa)	Lisboa	Sim
Universidade da Madeira (UMa)	Funchal	Sim
Universidade do Minho (UM)	Braga e Guimarães	Sim
Universidade Nova de Lisboa (NOVA)	Lisboa	Sim

Universidade do Porto (UPorto)	Porto	Sim

Tabela 5 - Principais institutos politécnicos públicos de Portugal

Politécnico	Localização	Plano curricular
Instituto Politécnico de Beja (IPBeja)	Beja	Sim
Instituto Politécnico de Bragança (IPB)	Bragança	Sim
Instituto Politécnico de Castelo Branco (IPCB)	Castelo Branco	Não
Instituto Politécnico do Cávado e Ave (IPCA)	Barcelos	Sim
Instituto Politécnico de Coimbra (IPC)	Coimbra	Sim
Instituto Politécnico da Guarda	Guarda	Sim
Instituto Politécnico de Leiria (IPLeiria)	Leiria	Não
Instituto Politécnico de Lisboa (IPL)	Lisboa	Sim
Instituto Politécnico de Portalegre (IPPt)	Portalegre	Não
Instituto Politécnico do Porto (IPP)	Porto	Não
Instituto Politécnico de Santarém (IPSantarém)	Santarém	Não
Instituto Politécnico de Setúbal (IPS)	Setúbal	Sim
Instituto Politécnico de Tomar (IPT)	Tomar	Sim
Instituto Politécnico de Viana do Castelo (IPVC)	Viana do Castelo	Sim
Instituto Politécnico de Viseu (IPV)	Viseu	Sim

Tabela 6 - Conteúdos abordados nas principais universidades públicas

Universidade	Conteúdos
IST	**Programação**: Programação imperativa, *C.* **Análise de algoritmos**: Eficiência de algoritmos, melhor, pior e caso médio. **Algoritmos de ordenação:** *Insertion sort, selection sort, bubble sort, quick sort, merge sort, heap sort, shell sort, counting sort, radix sort.* **Estruturas de dados:** Pilhas, filas, filas de prioridade, *heap*, árvores, árvore binária de pesquisa, árvore de pesquisa balanceada e implementações vetoriais e dinâmicas. **Tabelas de dispersão:** Resolução de colisões por encadeamento e por endereçamento aberto, endereçamento linear, quadrático e dispersão dupla.
ISCTE	**Análise de algoritmos***:* Problema *Union-Find,* melhor, pior e caso médio.

	Algoritmos de ordenação: *Selection sort, insertion sort, shell sort, merge sort, quick sort, heap sort.* **Estruturas de dados:** Pilhas, filas, listas, *bags*, filas de prioridade, árvore de pesquisa balanceada. Tabelas de dispersão e tabelas de símbolos elementares.
UAlg	**Programação**: Programação orientada a objetos*, Java*. **Análise de algoritmos:** melhor, pior e caso médio. **Algoritmos de ordenação:** *Insertion sort, selection sort, merge sort, quick sort.* **Estruturas de dados:** *Bags*, pilhas, filas, filas de prioridade, árvore binária de pesquisa, árvore de pesquisa balanceada. **Grafos:** Procura em profundidade, procura em largura, árvore geradora mínima, caminho mais curto. **Paradigmas e tópicos avançados:** Dividir para conquistar, algoritmos gananciosos, programação dinâmica, algoritmos sobre cadeias de *strings*. Tabelas de dispersão.
UA	**Programação**: *C* e *C++,* ponteiros*,* sobrecarga de funções, *templates*, STL. **Análise de algoritmos***:* Notação de complexidade, melhor, pior e caso médio. **Algoritmos de ordenação:** *Merge sort, quick sort, heap sort, etc.* **Estruturas de dados:** Pilhas, filas, deques, listas ligadas, árvore binária de pesquisa, árvore AVL, filas de prioridade. **Grafos:** Representação, algoritmos de travessia. **Paradigmas e tópicos avançados:** Diminuir e conquistar, dividir para conquistar, programação dinâmica, algoritmos determinísticos vs não determinísticos, problemas P, NP, NP-completos*,* algoritmos de aproximação. **Tabelas de dispersão:** Funções *hash*, endereçamento aberto, encadeamento separado.
UAc	**Análise de algoritmos:** Critérios de eficiência*,* notação *Big-O,* melhor, pior e caso médio. **Algoritmos de ordenação:** *Insertion sort, selection sort, bubble sort, shell sort, merge sort, quick sort, radix sort, heap sort.* **Estruturas de dados:** Pilhas, filas, listas, árvore binária de pesquisa, grafos, dicionários, implementações vetoriais e dinâmicas. **Grafos:** Caminho mais curto e custo mínimo.
UBI	**Análise de algoritmos:** Notação *Big-O,* análise amortizada*,* melhor, pior e caso médio. **Algoritmos de ordenação:** Radix sort. **Estruturas de dados***:* Listas ligadas, pilhas, filas, árvore binária de pesquisa, árvore AVL, árvore *red-black*, árvore 2-3, árvore de intervalos. **Filas de prioridade:** *Heap* binário*, heap* binomial*, heap* de Fibonacci. **Grafos:** Tipos, representação, procura em profundidade, procura em largura, árvore geradora mínima, caminho mais curto, fluxo máximo, emparelhamento estável.

	Paradigmas e tópicos avançados: Conjuntos de *strings*, KMP, *Boyer-Moore*, árvore de prefixos, árvore de sufixos, *array* de sufixos.
UC	**Análise de algoritmos:** Complexidade, técnicas gerais de *design* de algoritmos, melhor, pior e caso médio. **Algoritmos de ordenação:** *Shell sort, merge sort, quick sort, radix sort, MSD, LSD.* **Estruturas de dados:** Árvore AVL*, árvore* VP*, skip lists,* árvore B. **Paradigmas e tópicos avançados:** Mapeamento de *strings*.
UEvora	**Análise de algoritmos:** Complexidade temporal e espacial, notação *Big-O,* melhor, pior e caso médio. **Algoritmos de ordenação**: *Bubble sort, insertion sort, merge sort, heap sort, quick sort, bucket sort.* **Estruturas de dados:** Listas, pilhas, filas, árvore binária de pesquisa, árvore AVL. **Filas de prioridade:** *Heap* binário. **Tabelas de dispersão:** Funções *hash,* resolução de colisões por encadeamento, endereçamento aberto*, re-hashing.*
ULisboa	**Programação**: Recursividade, iteradores, génericos. **Análise de algoritmos:** Complexidade assintótica, melhor, pior e caso médio. **Algoritmos de ordenação:** *Insertion sort, merge sort, quick sort, heap sort.* **Estruturas de dados:** *Bags*, filas, pilhas, árvore binária de pesquisa, árvore *red-black*. Tabelas de dispersão
UMa	**Programação**: *C++,* ponteiros, memória dinâmica. **Análise de algoritmos:** Eficiência, melhor, pior e caso médio. **Algoritmos de ordenação:** *Insertion sort, selection sort, bubble sort, shell sort, heap sort, quick sort, merge sort.* **Estruturas de dados:** Pilhas, filas, listas ligadas, árvore n-área, árvore binária de pesquisa, árvore de pesquisa balanceada, árvore B. **Grafos:** Representação, algoritmos básicos. **Tabelas de dispersão:** *Hashing*, pesquisa, inserção, remoção.
UM	**Análise de algoritmos:** Análise da correção de algoritmos, complexidade assintótica, melhor, pior e caso médio. **Estruturas de dados:** Árvore AVL, *heap*, tabelas de dispersão. **Grafos:** Percurso em grafos, caminho mais curto, árvore geradora mínima. **Paradigmas e tópicos avançados:** Dividir para conquistar, algoritmos gananciosos, programação dinâmica, problemas NP-completos.
UPorto	**Análise de algoritmos:** Análise da correção de algoritmos, complexidade assintótica, melhor, pior e caso médio. **Algoritmos:** Algoritmos de ordenação e pesquisa. **Estruturas de dados:** Pilhas, filas, listas, árvore binária de pesquisa, árvore de pesquisa balanceada, tabelas de dispersão, filas de prioridade.

	Grafos: Procura em profundidade, procura em largura, ordenação topológica, deteção de ciclos, conectividade.
NOVA	**Análise de algoritmos:** Complexidade temporal e espacial, melhor, pior e caso médio. **Algoritmos de ordenação:** *Insertion sort, selection sort, bubble sort, heap sort, merge sort, quick sort, bucket sort, radix sort.* **Estruturas de dados:** Pilhas, filas, listas, dicionários, filas de prioridade, tabelas de dispersão, árvore genérica, árvore binária de pesquisa, árvore AVL, *heap*. **Paradigmas e tópicos avançados:** Dividir para conquistar, memorização.

Tabela 7 - Conteúdos abordados nos principais politécnicos públicos

Politécnico	Conteúdos
IPBeja	**Análise de algoritmos:** Complexidade computacional, técnicas gerais de *design* de algoritmos, equações de recorrência, teorema mestre. **Algoritmos de ordenação:** Problema da ordenação. **Estruturas de dados:** Conjuntos dinâmicos, árvore binária de pesquisa, árvore de pesquisa balanceada. **Grafos:** Procura em profundidade, procura em largura, árvore geradora mínima, caminho mais curto. **Tabelas de dispersão:** Mapas associativos.
IPB	**Programação:** *Java,* programação orientada a objetos. **Análise de algoritmos:** Eficiência do algoritmo, funções de referência, análise assintótica, notação *Big-O*. **Algoritmos de ordenação:** Métodos de ordenação das coleções *Java*. **Estruturas de dados:** Listas ligadas, pilhas, filas, deques, dicionários, árvore binária de pesquisa, árvore AVL, filas de prioridade.
IPCA	**Programação:** C, Ponteiros. **Estruturas de dados:** Listas ligadas, tabelas de dispersão, árvore binária de pesquisa, grafos.
IPC	**Programação:** *Java,* programação genérica. **Análise de algoritmos:** Análise de complexidade . **Estruturas de dados:** Pilhas, filas, árvore de Huffman, árvore binária de pesquisa, árvore *red-black*, árvore B, árvore AVL, árvore *splay*, avaliação de expressões infixas e pós-fixas, estruturas de dados multidimensionais. **Tabelas de dispersão:** Probing linear e quadrático, *r*esolução de colisões por encadeamento.
IPG	**Programação:** Linguagem algorítmica, fluxogramas (*flowcharts*), pseudocódigo, tipos de variáveis, instruções de *input* e *output*, estruturas de controlo (*control structures*).

	Análise de algoritmos: Complexidade temporal e espacial, notação *Big-O (Big-O notation),* notação *Little-o (Little-o notation),* crescimento de funções. **Algoritmos de ordenação:** Pesquisa sequencial e binária. **Estruturas de dados:** *Strings, ficheiros, arrays, matrizes,* listas ligadas (*linked lists),* filas (queues), pilha (*stacks*), árvores (*trees).*
IPL	**Algoritmos de ordenação:** *Insertion sort, selection sort, bubble sort, quick sort, merge sort, heap sort; linear-time sorting algorithms.* **Estruturas de dados:** Pilhas (*stacks*), filas (*queues*), filas de prioridade (*priority queues*), *heap*, árvore binária de pesquisa (*binary search tree*), conjuntos disjuntos (*disjoint sets*), grafos (graphs), arrays, listas ligadas (*linked lists*), tabelas de dispersão (*hash tables*), listas / matrizes de adjacência. **Grafos:** Algoritmos de pesquisa em grafos. **Paradigmas e tópicos avançados:** técnicas para projetar e analisar estruturas de dados e algoritmos
IPSantarém	**Programação:** pseudocódigo, Recursividade, *MVC.* **Algoritmos:** Pesquisa, seleção e ordenação. **Análise de algoritmos:** Notação *Big-O*, melhor, pior e caso médio. **Estruturas de dados:** Listas, árvores, árvores de decisão, *pilhas,* filas, implementações estáticas, dinâmicas, recursivas e não recursivas.
IPS	**Programação:** *C*, ponteiros. **Análise de algoritmos:** Complexidade algorítmica, notação *Big-O.* **Algoritmos de ordenação:** *Bubble sort, selection sort, quick sort,* algoritmos recursivos vs iterativos. **Estruturas de dados:** Pilhas, filas, listas, mapas, *arrays*, listas ligadas. **Tabelas de dispersão :** *ADT Map,* tabelas de dispersão.
IPT	**Programação:** Técnicas de desenvolvimento de algoritmos **Análise de algoritmos:** Análise de complexidade. **Estruturas de dados:** Estruturas de dados lineares e hierárquicas.
IPVC	**Programação:** C, fluxo de controlo, *arrays*, *array* multidimensional, funções, *strings*, estruturas, ponteiros. **Análise de algoritmos:** Eficiência algorítmica. **Estruturas de dados:** Pilhas, filas, listas ligadas, árvores.

4.3.1 Critérios específicos de inclusão

Após a análise dos dados recolhidos, procedeu-se à definição dos algoritmos a integrar na primeira versão funcional do *Algoriddle*. Esta decisão teve em conta quatro dimensões essenciais para a adequação didática e tecnológica da aplicação. Considerou-se a capacidade de cada

algoritmo para apoiar a compreensão de conceitos estruturantes, tais como iteração, manipulação de dados e análise de desempenho, facilitando a interpretação dos processos subjacentes. Avaliou-se também o potencial de visualização dos seus passos internos, permitindo representar, de forma clara e animada, a evolução dos valores e das estruturas utilizadas durante a execução.

A relevância prática dos algoritmos na formação académica e no contexto profissional foi igualmente tida em consideração, assegurando que os conteúdos disponibilizados na plataforma possuíssem aplicabilidade reconhecida em múltiplos cenários de resolução de problemas. Finalmente, procedeu-se à verificação da viabilidade técnica da implementação de cada algoritmo, garantindo que a solução pudesse manter um funcionamento estável e fluido, em coerência com os requisitos tecnológicos definidos para o sistema.

A decisão final beneficiou também do acompanhamento do orientador e da reflexão do autor sobre os conteúdos que poderiam melhor contribuir para a experiência de aprendizagem pretendida, sempre em conformidade com os resultados obtidos na análise curricular anteriormente descrita.

Assim, e como ilustrado na Tabela 8, foram selecionados algoritmos pertencentes às categorias de ordenação, pesquisa e grafos, que constituem o núcleo essencial de competências abordadas nas unidades curriculares alvo desta investigação.

Tabela 8 - Proposta de algoritmos a serem implementados

Categoria	Algoritmos
Ordenação	*Bubble sort, insertion sort, selection sort, merge sort, quick sort, counting sort, bucket sort.*
Procura	Pesquisa sequencial, pesquisa binária.
Algoritmos em Grafos	Procura em profundidade (DFS), procura em largura (BFS)

4.4 Arquitetura e *design* do sistema

A arquitetura do *Algoriddle* foi concebida com o objetivo de assegurar estabilidade técnica, escalabilidade e flexibilidade para futuras evoluções do sistema. Atendendo às necessidades identificadas na fase de requisitos e às boas práticas de engenharia de *software*, optou-se pela implementação de uma arquitetura *web* modular assente na separação de responsabilidades entre a camada de apresentação, a camada de controlo lógico e o serviço dedicado à execução de código. Esta distribuição permite reduzir a complexidade interna do sistema, reforçar a

segurança durante o processamento e facilitar a manutenção ou extensão das funcionalidades ao longo de novos ciclos de desenvolvimento (Figura 11).

A camada de *interface* (*frontend*) foi implementada recorrendo ao *framework React*, aliado a *TypeScript* e ao *bundler Vite*, proporcionando uma estrutura robusta na organização dos componentes e garantindo uma interação fluida mesmo quando o utilizador manipula sequências complexas de execução algorítmica. Paralelamente, o *Node.js* com *Express* foi utilizado como base para o *backend*, encarregando-se de gerir pedidos, coordenar fluxos de comunicação e aplicar validações essenciais ao correto funcionamento da aplicação, conforme definido nas especificações técnicas anteriormente apresentadas .

Dada a natureza sensível da execução de código submetido pelo utilizador, o processamento é realizado externamente através da *Piston API,* num ambiente isolado que previne interferências indesejadas com o servidor principal e impede a execução de instruções potencialmente perigosas. Este mecanismo de isolamento fortalece a segurança do sistema e garante que os utilizadores podem experimentar diferentes cenários de execução sem comprometer a integridade da plataforma. Em conjunto, estas decisões arquitetónicas refletem a preocupação em conciliar desempenho, proteção e modularidade, assegurando um enquadramento tecnológico compatível com os objetivos educativos do projeto.

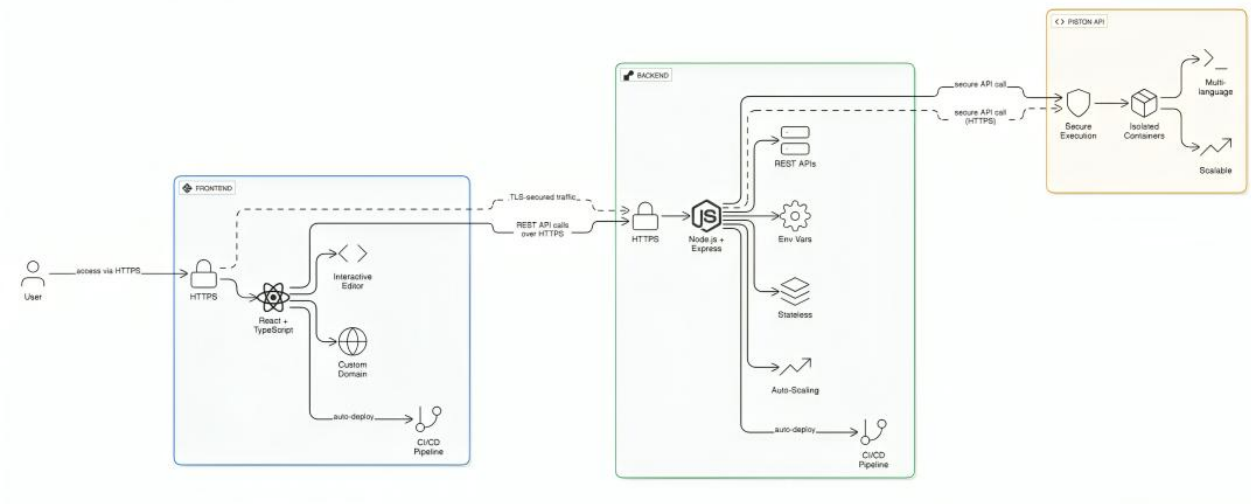


Figura 11 - Arquitetura geral do sistema

4.5 Usabilidade e *design* inclusivo

O *design* do *Algoriddle* foi orientado por princípios de usabilidade, acessibilidade e *design* inclusivo, com o objetivo de proporcionar uma experiência de aprendizagem simples, intuitiva e funcional, reduzindo a carga cognitiva associada à compreensão de algoritmos. A conceção da

interface seguiu as orientações da norma ISO 9241-210:2019, relativa ao *design* centrado no utilizador, privilegiando a clareza visual, a consistência gráfica e a previsibilidade da interação (International Organization for Standardization, 2019). A organização da *interface* promove uma separação clara entre código, visualização e informação teórica, evitando a apresentação simultânea de elementos excessivos que possam distrair ou sobrecarregar o utilizador.

A redução da carga cognitiva foi abordada através da decomposição das interações em passos controlados, do destaque visual do estado atual do algoritmo e do controlo explícito do ritmo de execução. O uso de *feedback* imediato e formativo permite ao utilizador compreender o impacto das suas ações, apoiar o raciocínio lógico e facilitar a construção de modelos mentais consistentes, em conformidade com os princípios da Teoria da Carga Cognitiva, nomeadamente na minimização da carga extrínseca (Sweller, 2011). A simplicidade visual e a coerência na disposição dos elementos contribuem para uma experiência de utilização que favorece a compreensão de conceitos abstratos e o envolvimento ativo do utilizador.

Em termos de acessibilidade e *design* inclusivo, foram considerados princípios como contraste adequado, legibilidade tipográfica, navegação previsível, consistência visual, compatibilidade com diferentes dispositivos e possibilidade de personalização, incluindo modos claro e escuro. Estas decisões estão alinhadas com as recomendações das Web Content Accessibility Guidelines (WCAG 2.1, nível AA) do W3C/WAI (World Wide Web Consortium, 2018), ainda que a versão atual do *Algoriddle* não integre funcionalidades avançadas de apoio a utilizadores invisuais ou com limitações sensoriais específicas. Nesta fase, a acessibilidade é sobretudo assegurada através da simplicidade de interação e da usabilidade geral da aplicação.

Adicionalmente, o *Algoriddle* incorpora princípios de *design* inclusivo inspirados no Universal *Design* for *Learning* (UDL), ao permitir múltiplas formas de representação e envolvimento (UNESCO, 2017). A visualização animada dos algoritmos, sincronizada com o código, facilita a articulação entre a lógica abstrata e o comportamento concreto do programa, promovendo uma aprendizagem ativa, flexível e adaptada a diferentes perfis cognitivos e níveis de experiência em programação. A combinação entre clareza visual, *feedback* imediato, suporte multilingue e personalização estabelece uma base sólida para futuras evoluções da plataforma, orientadas pelos princípios do *design* universal e da educação inclusiva.

4.6 Modelagem e estrutura do sistema

A modelagem funcional do *Algoriddle* foi realizada com base numa abordagem centrada no utilizador, representando todas as interações possíveis entre este e o sistema. O utilizador é considerado o ator principal, podendo selecionar e executar algoritmos, controlar a sua execução (pausar, avançar ou reiniciar), visualizar o processo de forma animada, importar dados personalizados e consultar informação teórica adicional. O sistema atua como ator auxiliar,

processando o código submetido, executando os algoritmos em ambiente isolado, atualizando a visualização e apresentando *feedback* estruturado.

Para documentar graficamente estas interações, foram criadas duas representações distintas. Primeiro, um diagrama de casos de uso (Figura 12), que sintetiza os requisitos funcionais do sistema e ilustra os serviços disponibilizados aos utilizadores. Em paralelo, um diagrama BPMN (Figura 13) que descreve o fluxo de interação e os processos internos do *Algoriddle*, detalhando a sequência de atividades desde a submissão do código até à apresentação dos resultados e *feedback*.

Estas representações proporcionam uma visão consolidada do funcionamento do sistema, facilitando a documentação e servindo de base para futuras evoluções ou integrações. A utilização de diagramas estruturados garante que os requisitos funcionais são cumpridos de forma consistente e clara, reforçando a eficácia pedagógica e a experiência do utilizador.

Figura 12 - Diagrama de casos de uso da aplicação

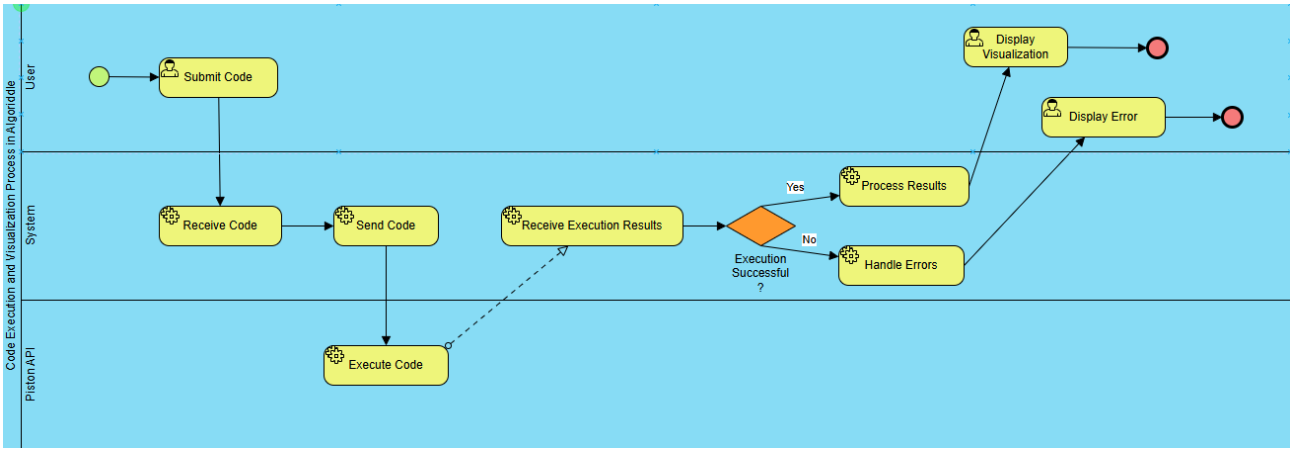


Figura 13 - Diagrama BPMN do fluxo de execução de código

4.7 Fluxo de interação e experiência do utilizador

O fluxo de interação no *Algoriddle* foi concebido de forma a assegurar uma experiência simples, eficiente e alinhada com os objetivos pedagógicos da plataforma, garantindo que o utilizador dispõe de um percurso de uso lógico e contínuo desde a escrita do código até à observação animada da sua execução. A estrutura definida para esta experiência corresponde a um ciclo de comunicação constante entre os componentes do sistema, conforme descrito previamente na arquitetura e modelagem funcional, mantendo a coerência com os princípios de *design* inclusivo e usabilidade adotados durante o desenvolvimento do artefacto.

4.7.1 Ciclo de interação

O ciclo de interação inicia-se quando o utilizador seleciona ou introduz um algoritmo no editor de código, num ambiente orientado para facilitar a escrita e edição por estudantes em fases iniciais de aprendizagem. Após a submissão, o *frontend* envia o código e os respetivos parâmetros ao *backend*, onde decorre um processo de validação e tratamento seguro dos dados, assegurando a integridade da execução. Seguidamente, o código é transmitido para um serviço de execução isolado, que impede interferências no funcionamento geral da aplicação e garante a segurança do servidor .

A execução produz três tipos de resultados principais: saída normal, mensagens de erro e rastreamento da execução passo a passo, permitindo observar o comportamento interno dos algoritmos. A resposta é devolvida ao *frontend*, que organiza a informação e a apresenta de forma visual e animada, permitindo ao utilizador controlar o ritmo de exploração. Este processo

é representado no diagrama de sequência da Figura 14, ilustrando a comunicação contínua entre os componentes do sistema.

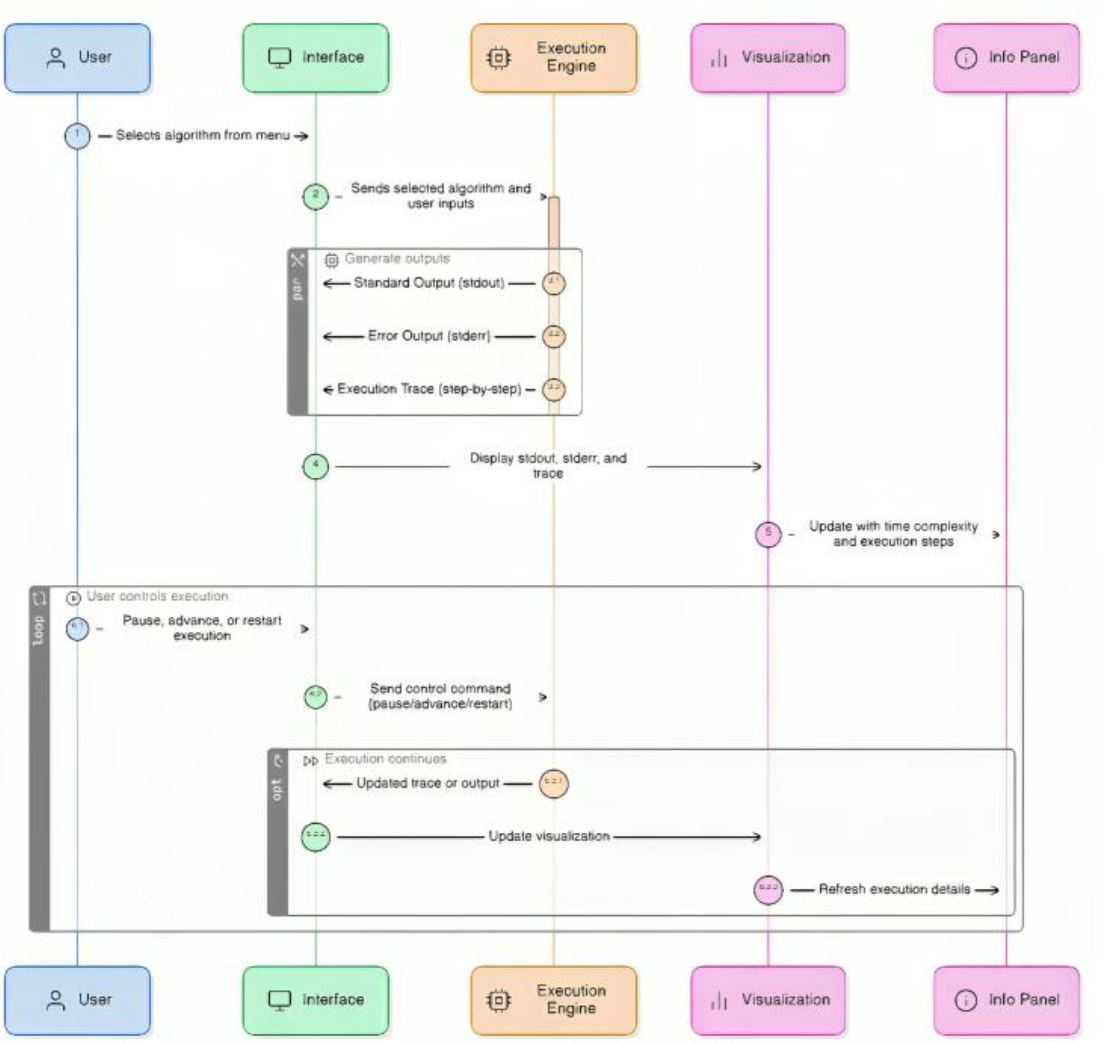


Figura 14 - Diagrama de sequência do ciclo de interação entre componentes

4.7.2 *Interface* do utilizador

O *Algoriddle* está disponível online e pode ser acedido através do endereço https://algoriddle.netlify.app/. Esta versão reflete o artefacto avaliado na presente pesquisa, integrando as decisões de *design*, interação e implementação descritas neste capítulo.

A *interface* gráfica desempenha um papel central na experiência de utilização, funcionando como mediadora entre o utilizador e a execução visual dos algoritmos. O *design* da *interface* privilegia simplicidade, clareza e acessibilidade, reduzindo a carga cognitiva extrínseca e facilitando a compreensão de conceitos frequentemente abstratos no ensino da programação. Essa abordagem está alinhada com os princípios adotados no desenvolvimento do sistema, que

procuram promover interações intuitivas e coerentes com diferentes perfis de utilizadores. Como ilustrado na Figura 15, a *interface* foi projetada para ser facilmente navegável, criando uma experiência de aprendizagem mais fluida e acessível.

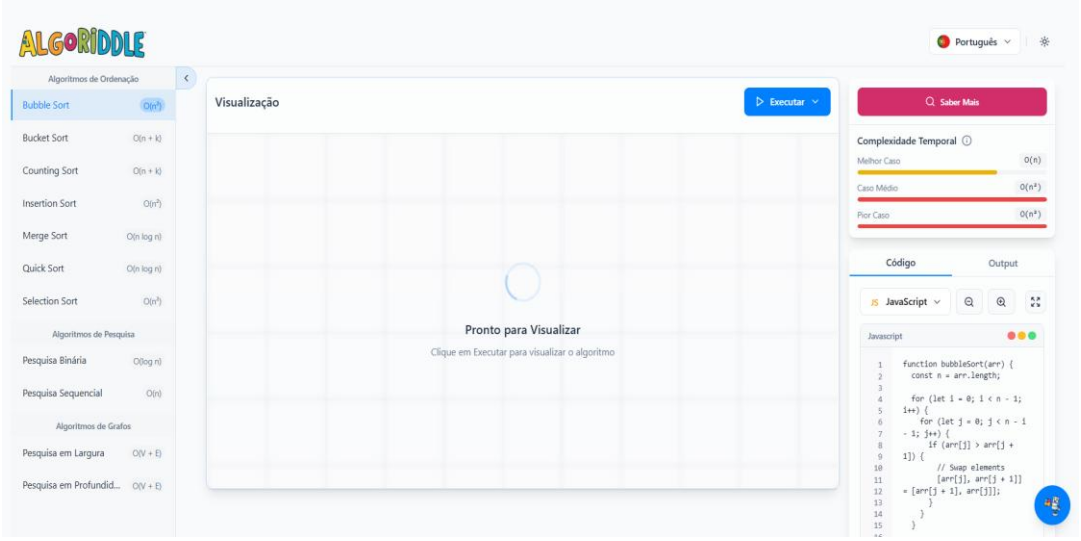


Figura 15 - Ecrã principal

O primeiro ponto de contacto com a aplicação é o editor de código, disponibilizado no ecrã principal, conforme mostrado na Figura 16. Neste módulo, o utilizador tem a possibilidade de escrever, editar e submeter algoritmos que são suportados pela plataforma. A escolha de um editor claro e responsivo visa reduzir erros comuns e apoiar especialmente aqueles que estão nas etapas iniciais da educação em computação, garantindo que a parte técnica não se sobreponha à aprendizagem dos conceitos. Esse *design* intuitivo facilita o processo de aprendizagem de programação, criando um ambiente mais acessível para iniciantes.

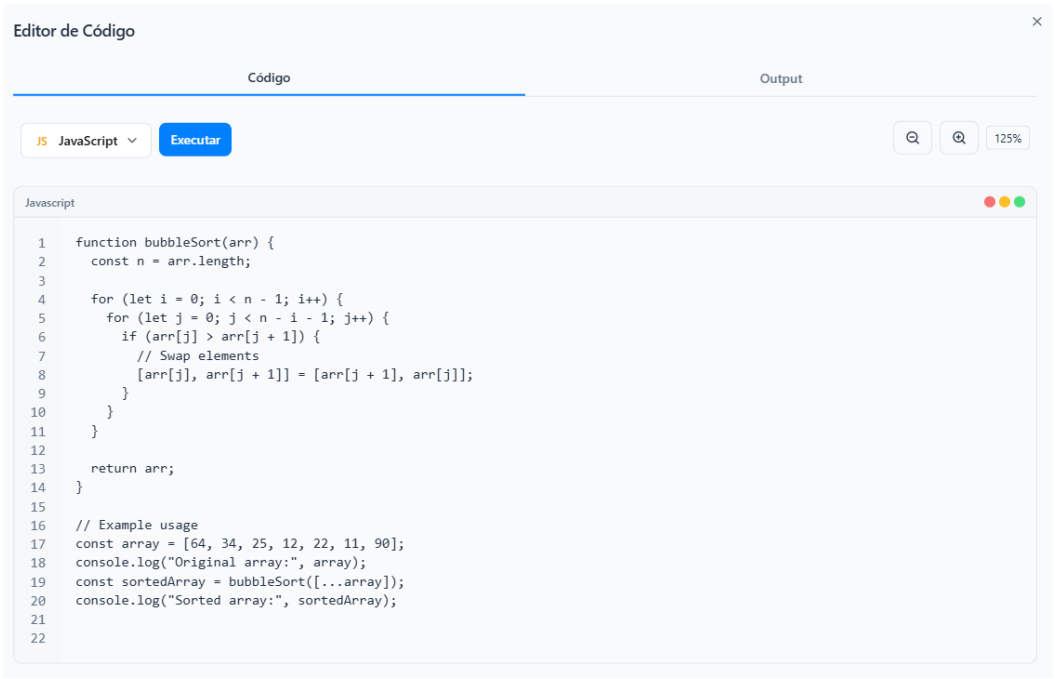


Figura 16 - Ecrã do editor de código

Após a submissão, o algoritmo é executado e os seus passos tornam-se observáveis no visualizador interativo, conforme ilustrado na Figura 17. Este componente permite ao utilizador observar passo a passo o comportamento do algoritmo, oferecendo a possibilidade de controlar o ritmo da execução através de ações como pausar, avançar ou reiniciar. Essa interação favorece uma interpretação progressiva das operações internas, permitindo uma melhor compreensão do fluxo de execução e facilitando a construção mental do raciocínio algorítmico.

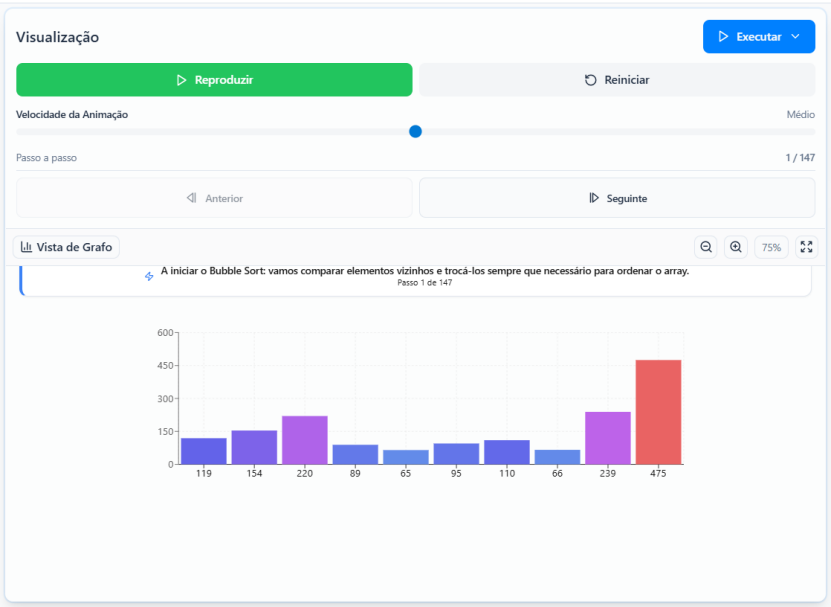


Figura 17 - Ecrã do visualizador

Para reforçar a conexão entre teoria e prática, o sistema disponibiliza um painel informativo que apresenta dados essenciais sobre o algoritmo em análise, incluindo a complexidade temporal, conforme ilustrado na Figura 18. Este painel fornece uma visão clara e direta das características algorítmicas, permitindo ao utilizador compreender rapidamente o desempenho do algoritmo em termos de tempo de execução. Além disso, são fornecidas descrições complementares que contextualizam o funcionamento do método e as suas aplicações mais comuns, como exemplificado na Figura 19. A organização da informação teórica visa apoiar uma compreensão gradual dos conceitos, ao mesmo tempo que incentiva a comparação fundamentada entre diferentes abordagens, facilitando o aprendizado e a análise crítica dos algoritmos.

Figura 18 - Ecrã do painel informativo

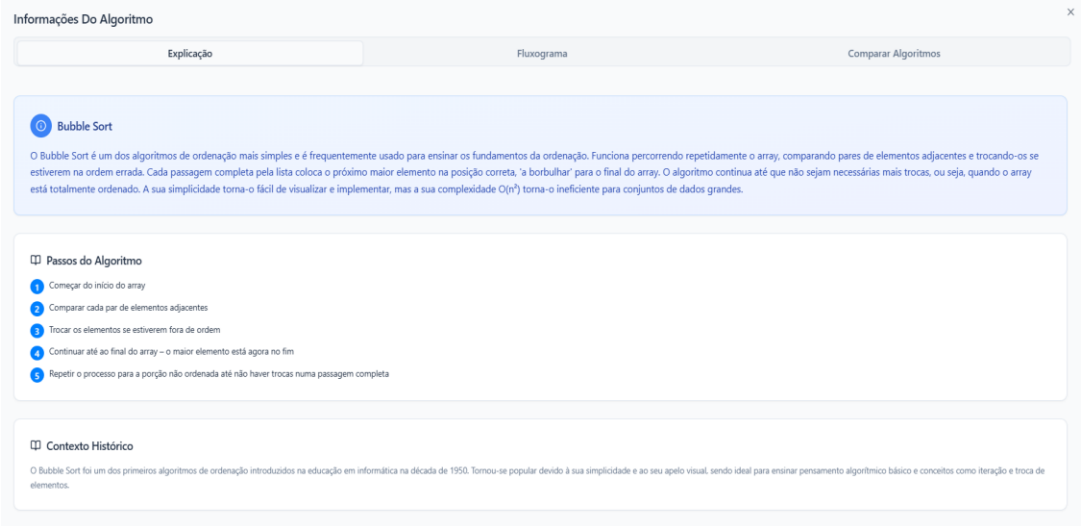


Figura 19 - Informações teóricas do algoritmo

Outro recurso integrado à aplicação é o fluxograma, que está presente nos algoritmos suportados e representa de forma visual e sequencial o fluxo de execução do algoritmo. Essa

representação gráfica facilita o entendimento da lógica subjacente ao algoritmo, especialmente em situações que envolvem ramos condicionais e ciclos. O fluxograma proporciona um mapeamento cognitivo mais sólido das operações realizadas, tornando mais intuitiva a compreensão dos passos envolvidos na execução do algoritmo. Como ilustrado na Figura 20, o fluxograma oferece uma visão clara e estruturada do processo, o que é particularmente útil para iniciantes ou para reforçar a compreensão de conceitos mais complexos.

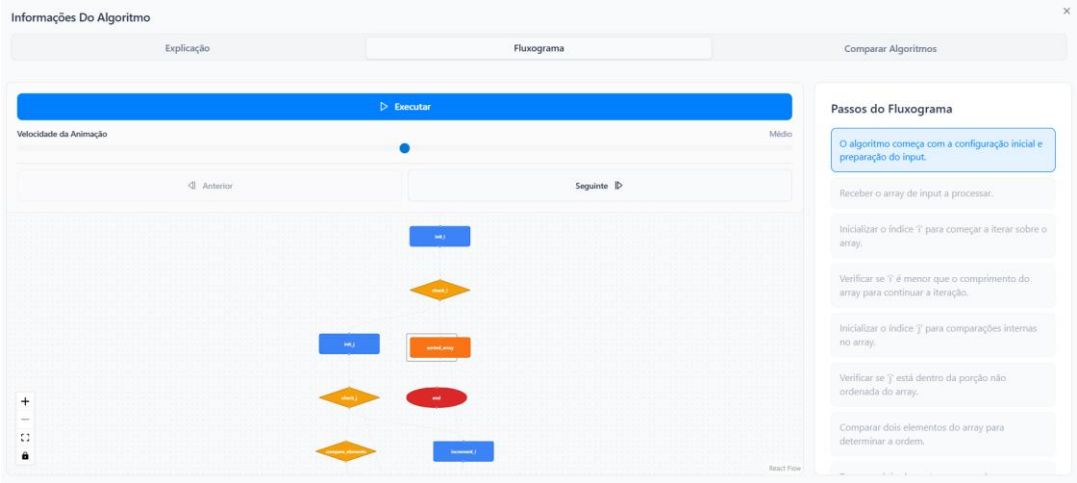


Figura 20 - Fluxograma do algoritmo

A aplicação também inclui um comparador de algoritmos, permitindo ao utilizador observar o comportamento de diferentes algoritmos em termos de complexidade temporal e espacial. Este comparador facilita a visualização das diferenças de desempenho, número de operações e eficiência algorítmica em cenários de melhor, pior e médio caso. Embora tenha sido inicialmente desenvolvido como uma ferramenta de apoio visual exploratório, o impacto pedagógico foi positivo, conforme evidenciado nas perceções recolhidas durante a avaliação do artefacto. Como ilustrado na Figura 21, o comparador permite uma análise detalhada das variações no desempenho dos algoritmos em diferentes condições, contribuindo para uma compreensão mais profunda da teoria por trás das operações algorítmicas.

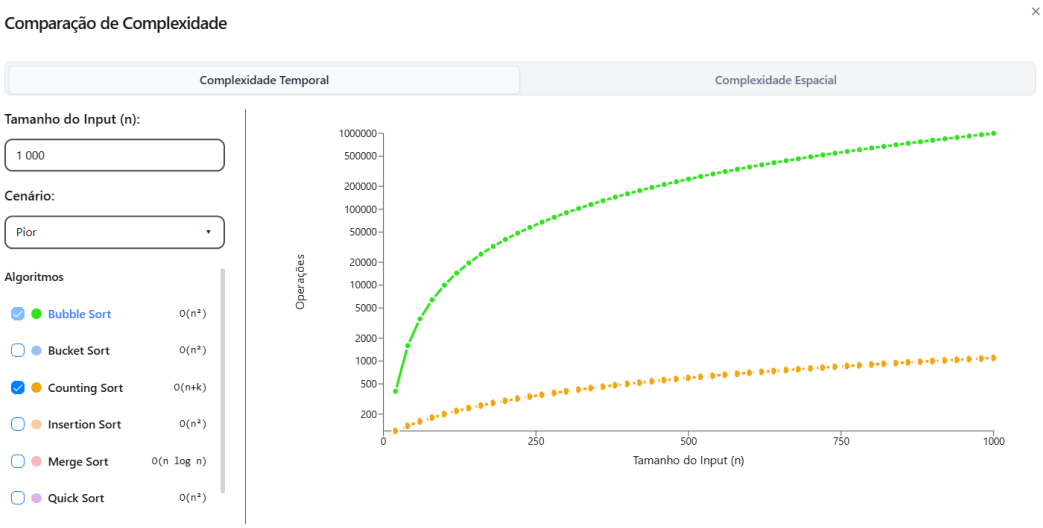


Figura 21 - Comparador de algoritmos

Por fim, o sistema integra um *chatbot* e uma área de perguntas frequentes (FAQs), que disponibilizam respostas rápidas e apoio conceptual em tempo real. Esses recursos permitem ao utilizador esclarecer dúvidas sem a necessidade de abandonar a plataforma, promovendo uma aprendizagem autónoma e personalizada. A Figura 22 exemplifica a apresentação desses componentes, que são projetados para fornecer suporte imediato e facilitar o processo de aprendizagem, ajustando-se às necessidades individuais de cada utilizador.

Figura 22 - Ecrã do *chatbot*

A combinação desses módulos resulta numa experiência coerente e centrada no utilizador, integrando exploração prática, visualização animada e suporte teórico imediato. Esta abordagem multifacetada garante que o utilizador não só compreenda os conceitos de forma teórica, mas também tenha a oportunidade de aplicar e visualizar os algoritmos em ação. Todo o processo de interação está igualmente documentado no manual do utilizador (Apêndice 1, página 65), que fornece instruções detalhadas e orientações para maximizar o potencial da plataforma, assegurando que os utilizadores possam tirar o melhor proveito de todas as funcionalidades disponibilizadas.

5 Avaliação

Este capítulo apresenta a avaliação sistemática da aplicação *Algoriddle*, realizada com o objetivo de analisar a sua usabilidade, pertinência pedagógica e contributo para a compreensão de algoritmos por parte dos seus utilizadores. Esta fase constitui um momento central no processo de investigação, pois confronta o artefacto com a sua utilização real e permite a recolha de perceções e dados sobre a experiência dos utilizadores com diferentes níveis de conhecimento em programação. A avaliação procurou fornecer evidências empíricas acerca da clareza da interação, da adequação das funcionalidades às necessidades identificadas e da qualidade global da experiência de utilização, permitindo assim avaliar o alinhamento entre os objetivos de desenvolvimento do artefacto e a perceção dos participantes.

5.1 Objetivos da avaliação

O principal objetivo da avaliação consistiu em compreender a eficácia da aplicação na promoção da aprendizagem de algoritmos, através da visualização animada e da interação orientada. Foram analisados diversos aspetos, incluindo a clareza e organização da *interface*, a facilidade de navegação, a utilidade dos controlos de execução e a compreensão proporcionada pelas explicações associadas aos processos algorítmicos. Adicionalmente, procurou-se avaliar o valor pedagógico percebido pelos participantes, nomeadamente se a aplicação favorece a construção de representações mentais mais robustas sobre o funcionamento interno dos algoritmos.

A avaliação do *Algoriddle* centrou-se predominantemente na experiência do utilizador e na usabilidade percebida, tendo em conta a natureza exploratória do estudo e os limites temporais e contextuais da investigação. Não sendo o artefacto avaliado em contexto formal de sala de aula, optou-se por recolher perceções relacionadas com clareza, facilidade de utilização, acessibilidade e utilidade pedagógica potencial, em vez de medir ganhos de aprendizagem de forma experimental. Esta opção está alinhada com a fase de avaliação prevista no modelo DSR Educacional, que privilegia a validação da adequação e qualidade do artefacto antes de estudos pedagógicos longitudinais mais aprofundados.

A avaliação foi conduzida por estudantes e profissionais da área da informática, garantindo uma perspetiva técnica e prática sobre a aplicação.

5.2 Procedimentos e instrumentos de recolha de dados

A avaliação contou com a participação de quinze indivíduos, distribuídos por oito iniciantes, cinco intermédios e dois avançados, permitindo assim a recolha de perceções provenientes de diferentes níveis de experiência. Em conformidade com as normas éticas definidas no enquadramento metodológico, a informação recolhida concentrou-se exclusivamente na experiência em programação, não incluindo outros dados sociodemográficos.

Os dados foram obtidos através de um guião de tarefas e de um questionário misto, composto por itens de resposta fechada em escala Likert (1 a 5) e questões abertas, que orientaram os participantes na exploração das funcionalidades da aplicação. Cada tarefa incluía instruções específicas e espaço para comentários qualitativos, permitindo a recolha de perceções detalhadas sobre a clareza, utilidade e fluidez da interação. Adicionalmente, foi aplicado um questionário *System Usability Scale* (SUS) autónomo (Apêndice 4, página 109), de forma a obter uma métrica padronizada de usabilidade, dado que o guião original não contemplava todos os itens exigidos para o cálculo do SUS segundo (Brooke, 1996).

O tempo despendido por cada participante foi registado e ilustrado na Figura 23, apresentando uma média de 18 minutos e 34 segundos para a conclusão das tarefas e do questionário. Embora este resultado indique que os participantes conseguiram completar as tarefas de forma contínua e sem dificuldades significativas, deve ser interpretado com cautela, uma vez que o tempo relativamente reduzido poderá refletir uma exploração menos aprofundada de funcionalidades mais avançadas da aplicação.

Para a avaliação da usabilidade, foi utilizada a versão original do SUS, proposta por Brooke (1996), composta por 10 itens com respostas em escala Likert de cinco pontos. Apesar de existirem adaptações e extensões do SUS propostas por diferentes autores ao longo do tempo, optou-se pela versão original por se tratar de um instrumento amplamente validado, neutro em termos de domínio e adequado a estudos com amostras reduzidas.

As variações existentes na literatura mantêm a estrutura base do questionário, diferindo sobretudo na interpretação dos resultados ou em adaptações contextuais. Neste trabalho, foi seguida a formulação clássica do SUS, assegurando comparabilidade com estudos prévios e clareza metodológica. Não foram utilizadas versões alternativas ou instrumentos complementares por forma a manter a simplicidade do processo de avaliação e reduzir a carga cognitiva associada à resposta dos participantes.

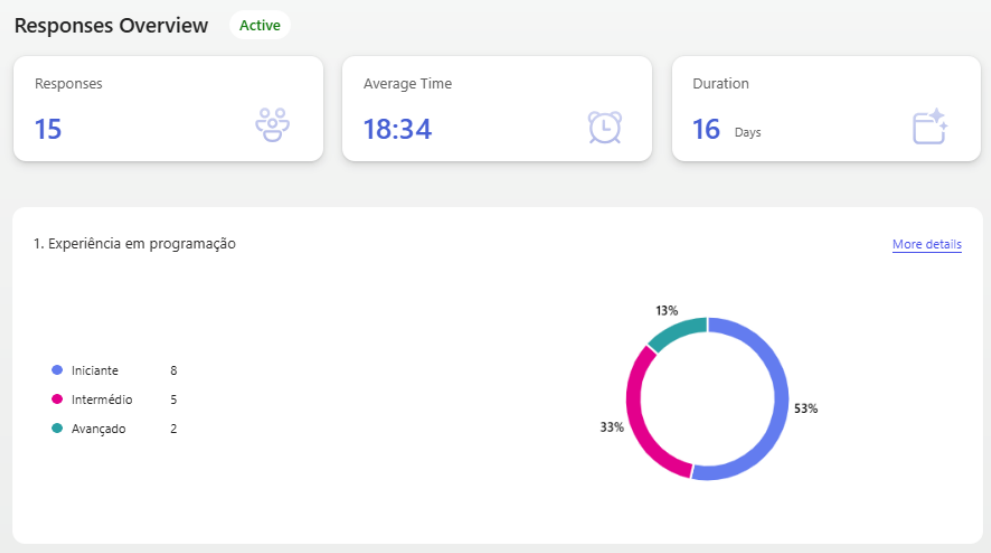


Figura 23 - Dados do questionário

5.3 Resultados

A análise dos dados recolhidos permitiu avaliar a aplicação *Algoriddle* em múltiplas dimensões, incluindo usabilidade, clareza funcional, adequação pedagógica e desempenho nas tarefas realizadas pelos participantes. Os resultados obtidos devem ser interpretados como perceções subjetivas relativas à experiência de utilização, não constituindo evidência direta de aprendizagem ou eficácia pedagógica, mas fornecendo indicadores relevantes para compreender o grau de aceitação, identificar potenciais limitações e orientar melhorias futuras.

O questionário aplicado estava organizado em várias secções, permitindo uma análise segmentada, tarefas, usabilidade, impacto pedagógico, funcionalidades de apoio e recomendação, cujos resultados quantitativos são apresentados e discutidos de seguida.

5.3.1 Resultados quantitativos

As avaliações das tarefas realizadas pelos participantes demonstram uma perceção maioritariamente positiva da aplicação. As tarefas iniciais, relacionadas com navegação, estrutura visual e interpretação dos elementos principais, registaram classificações entre 3 e 5, com médias em torno de 4.1. Estes valores sugerem que a interface foi considerada clara e funcional, embora a variabilidade observada indique espaço para otimizações incrementais.

As tarefas ligadas à execução e visualização de algoritmos obtiveram classificações predominantemente entre 4 e 5, revelando que os mecanismos de animação, controlo da

execução e manipulação de dados foram bem compreendidos pelos participantes. As tarefas associadas à construção e exploração de grafos e ao editor de código multilingue apresentam as médias mais elevadas (4.53), sugerindo que estas funcionalidades se destacaram pela clareza e utilidade percebida. Em contraste, a tarefa relativa ao apoio inteligente e FAQs obteve a média mais baixa (3.53), refletindo uma preferência por ferramentas externas mais avançadas, como o *ChatGPT*, para pesquisa imediata e resolução de dúvidas. A Tabela 9 sintetiza estes resultados.

Tabela 9 - Estatísticas das avaliações das tarefas

Tarefa	Média	Desvio padrão
1 - Navegação e estrutura da *interface*.	4.13	0.88
2 - Execução de um algoritmo de ordenação.	3.60	0.88
3 - Personalização de dados	4.27	0.77
4 - Comparação de algoritmos	4.00	0.89
5 -Execução de algoritmos de pesquisa	3.93	0.57
6 - Construção e Exploração de Grafos	4.53	0.62
7 - Editor de código multilingue	4.53	0.62
8 - Conteúdo didático e painel informativo	4.27	0.77
9 - Multilinguismo	4.40	0.80
10 - Apoio inteligente e FAQs	3.53	0.62

No que diz respeito ao questionário, as diferentes secções permitiram caracterizar as perceções associadas à usabilidade, às visualizações, às funcionalidades avançadas e ao potencial pedagógico do sistema. Na secção dedicada à usabilidade, os itens registaram médias entre **4.40** e **4.67**, com desvios-padrão reduzidos, indicando perceções tendencialmente favoráveis. A **visualização gráfica**, incluindo animações e legendas, obteve a média mais elevada (**4.67**), sugerindo que este componente foi particularmente bem avaliado. A aplicação também foi considerada fluida, com a média de **4.53** para a questão sobre **resposta sem atrasos**. Estes resultados detalhados podem ser consultados na Tabela 10.

Tabela 10 - Estatísticas da secção C (Impacto na usabilidade)

Item	Média	Desvio padrão
1 - A *interface* é intuitiva e de fácil utilização.	4.47	0.50
2- Os controlos de execução (pausa, retroceder, avançar, passo a passo) são claros e úteis.	4.40	0.49
3 - A navegação entre algoritmos e funcionalidades é rápida e consistente.	4.40	0.49
4 - As visualizações gráficas (animações e legendas) facilitam a compreensão.	4.67	0.47
5 - A aplicação respondeu de forma fluida e sem atrasos.	4.53	0.50

A análise da secção dedicada ao possível impacto pedagógico, apresentada na Tabela 11, revela que as visualizações passo a passo e as explicações teóricas foram percecionadas como elementos particularmente úteis e claros. As médias situaram-se entre 4.47 e 4.67, representando algumas das classificações mais elevadas de todo o questionário. Estes resultados indicam um forte alinhamento entre os objetivos pedagógicos da aplicação e a perceção dos participantes quanto ao valor das representações gráficas.

Tabela 11 - Estatísticas da secção D (Impacto na aprendizagem / pedagogia)

Item	Média	Desvio padrão
1- A visualização passo a passo contribuiu para melhor compreender os algoritmos.	4.53	0.50
2 - A possibilidade de usar dados personalizados tornou a experiência mais significativa.	4.47	0.50
3 - As explicações teóricas e complexidade foram úteis e rigorosas.	4.67	0.47
4 - A comparação entre algoritmos (melhor, médio, pior caso) ajudou a perceber diferenças de desempenho.	4.67	0.47
5 - As funcionalidades multilingues são úteis para aprendizagem e ensino em contextos diversos.	4.67	0.47

A análise dos dados apresentados na Tabela 12 sobre as funcionalidades de apoio revela uma avaliação mista dos recursos da aplicação. O *chatbot*, com uma média de 3.87, obteve a avaliação mais baixa, sugerindo que, embora tenha oferecido respostas claras e relevantes, há espaço para melhorias. Como mencionado anteriormente, essa pontuação pode indicar que os utilizadores preferem recorrer a ferramentas mais avançadas de pesquisa, como o *ChatGPT*, para obter respostas mais rápidas e precisas.

As FAQs e os *templates* de código em várias linguagens receberam uma média de 4.20, mostrando que foram considerados úteis para esclarecer dúvidas e apoiar a prática de programação. Já os conteúdos de apoio, como factos históricos e *flowcharts*, tiveram uma média de 4.13, sugerindo que, embora tenham sido úteis para reforçar o interesse e a aprendizagem, foram avaliados de forma um pouco mais moderada. Em geral, a secção de apoio teve uma avaliação positiva, com alguns recursos mostrando maior aceitação do que outros.

Tabela 12 - Estatísticas da secção E (Funcionalidades de apoio)

Item	Média	Desvio padrão
1- O *chatbot* forneceu respostas claras e relevantes.	3.87	0.88
2 - As FAQs são eficazes para esclarecer dúvidas frequentes.	4.20	0.75
3 - Os *templates* de código em diferentes linguagens apoiam a prática de programação.	4.20	0.83
4 - Os conteúdos de apoio (factos históricos, *flowcharts*) reforçam o interesse e a aprendizagem.	4.13	0.88

A análise da secção de recomendação e motivação, conforme apresentado na Tabela 13, revela que os participantes tiveram uma perceção positiva da aplicação. A média de 4.40 para a questão sobre motivação para explorar algoritmos indica que a aplicação foi eficaz em aumentar o interesse dos utilizadores pelos algoritmos. A média de 4.33 para a recomendação a outros estudantes sugere que os utilizadores consideram a aplicação útil o suficiente para indicá-la a colegas. Por fim, a maior média da secção, 4.67, foi atribuída à questão sobre o potencial de integração em contextos de ensino/aprendizagem, indicando que os utilizadores reconhecem o potencial da aplicação para ser utilizada em ambientes educativos.

Tabela 13 - Estatísticas da secção F (Recomendação e motivação)

Item	Média	Desvio padrão
1 - A aplicação aumentou a motivação para explorar algoritmos.	4.40	0.49
2 - Recomendaria esta aplicação a outros estudantes.	4.33	0.47
3 - Vejo potencial de integração da aplicação em contextos de ensino/aprendizagem.	4.67	0.47

A aplicação parcial da métrica SUS reforça estas perceções. Os resultados obtidos, apresentados na Tabela 14, demonstram que, após os devidos ajustes aos itens positivos e negativos, a soma total atingiu 33,5 pontos em 40 possíveis, correspondendo a uma pontuação final de 83,8 em 100 na escala SUS. Esta classificação insere o *Algoriddle* na categoria "excelente", de acordo com (Bangor et al., 2008), e confirma que o sistema é percebido como fácil de utilizar, coerente na integração das suas funcionalidades e capaz de proporcionar uma experiência de utilização confiável e fluida. A convergência entre esta métrica padronizada e as avaliações do questionário e das tarefas reforça a consistência dos resultados e a validade da avaliação.

Tabela 14 - Classificação SUS

Pergunta	Média (1-5)	Tipo	Score
1 - Acho que gostaria de usar este sistema com frequência.	4.7	Positivo	4,7 - 1 = 3,7
2 - Achei o sistema desnecessariamente complexo.	2.0	Negativo	5 - 2,0 = 3,0
3 - Achei o sistema fácil de usar.	4.8	Positivo	4,8 - 1 = 3,8
4 - Acho que precisaria de ajuda de alguém experiente para usar este sistema.	1.8	Negativo	5 - 1,8 = 3,2
5 - Achei que as várias funções do sistema estavam bem integradas.	4.6	Positivo	4,6 - 1 = 3,6
6 - Achei que havia demasiada inconsistência no sistema.	2,1	Negativo	5 - 2,1 = 2,9
7 - Imagino que a maioria das pessoas aprenderia a usar este sistema muito rapidamente.	4,5	Positivo	4,5 - 1 = 3,5
8 - Achei o sistema muito pesado de usar.	1,9	Negativo	5 - 1,9 = 3,1
9 - Senti-me bastante confiante a usar o sistema.	4,7	Positivo	4,7 - 1 = 3,7

10 - Necessitava de aprender muitas coisas antes de poder usar o sistema.	2,0	Negativo	5 - 2,0 = 3,0

5.3.2 Resultados qualitativos

A análise das respostas abertas do guião e questionário (Apêndice 2, página 80) permitiu aprofundar a interpretação dos dados quantitativos, oferecendo uma visão detalhada sobre a experiência dos participantes com a aplicação *Algoriddle*. Os utilizadores destacaram, de forma consistente, a relevância das funcionalidades de visualização e interatividade, nomeadamente as animações intuitivas e o modo passo-a-passo, apontados como recursos centrais para a compreensão dos algoritmos. O comparador foi igualmente valorizado, por permitir perceber claramente as diferenças de desempenho entre soluções, constituindo um elemento diferenciador da aplicação. Outras funcionalidades, como o editor multilingue e a criação e exploração de grafos, foram salientadas como úteis, reforçando a aplicabilidade prática da plataforma. O multilinguismo e a possibilidade de importar dados em formato JSON também se mostraram importantes para a adaptação da aplicação a diferentes contextos de aprendizagem, perceções que encontram suporte adicional na análise de conteúdo (Apêndice 3, página 96), onde a frequência e correlação de termos reforçam a consistência dos comentários recolhidos.

Os participantes valorizaram a *interface* e o *design*, considerando a organização visual clara e intuitiva, a apresentação agradável e a escolha adequada de cores. Destacou-se a fluidez da navegação, a disposição lógica dos painéis informativos e a clareza das explicações teóricas, que facilitaram a compreensão estruturada dos algoritmos. A suavidade das animações e a qualidade dos fluxogramas também foram frequentemente mencionadas, reforçando a perceção de uma experiência pedagógica e agradável.

Foram identificados alguns aspetos a melhorar, como maior fluidez no editor de código, tempos de execução mais rápidos e *feedback* mais explícito em operações como o *upload* de ficheiros. Os participantes sugeriram ainda exemplos práticos adicionais, legendas mais detalhadas e mensagens de erro mais visíveis, além de pequenas melhorias de *interface* que poderiam tornar a experiência mais contínua e satisfatória.

Alguns utilizadores propuseram funcionalidades adicionais com potencial pedagógico, incluindo modos de desafio, *quizzes*, vídeos curtos, secções de curiosidades, histórico de progresso e modos *offline*. Sugestões como *badges* de progresso e níveis de dificuldade graduais indicam que os participantes valorizam não só a clareza e utilidade da aplicação, mas também a motivação e o envolvimento dos utilizadores, aspetos essenciais para a aprendizagem autónoma.

Por fim, os comentários confirmam que a inovação da aplicação é reconhecida, sobretudo pelo comparador de algoritmos e pela ligação entre código e visualização, que facilitam a compreensão prática do funcionamento interno dos algoritmos. Em síntese, os resultados qualitativos mostram que os elementos visuais e interativos são determinantes para a perceção positiva da aplicação, enquanto os ajustes sugeridos representam oportunidades importantes para maximizar o impacto pedagógico em futuras versões.

6 Conclusões

O desenvolvimento e a avaliação do *Algoriddle* permitiram evidenciar o seu potencial enquanto ambiente digital de exploração visual e interativa de algoritmos, contribuindo para uma compreensão mais acessível de conceitos reconhecidamente abstratos especialmente para estudantes com menor experiência em programação. Os resultados obtidos demonstram que o artefacto cumpre o objetivo principal desta investigação que consistia em conceber, implementar e avaliar um ambiente virtual interativo para o ensino e aprendizagem de algoritmos, articulando princípios pedagógicos, design centrado no utilizador e rigor técnico, conforme definido no Capítulo 1.

A avaliação realizada confirmou que os objetivos iniciais foram alcançados. As visualizações, explicações e funcionalidades interativas foram amplamente percebidas como claras e eficazes na facilitação da compreensão dos processos algorítmicos. A usabilidade do sistema também se destacou, com uma pontuação de 83,8/100 no SUS, classificando a aplicação como "excelente". Além disso, as opiniões recolhidas sobre a utilidade pedagógica da aplicação foram altamente positivas. Dessa forma, pode-se concluir que a aplicação cumpre o seu propósito, mostrando um forte potencial para ser utilizada como um recurso educativo complementar.

6.1 Discussão dos resultados

Os resultados quantitativos e qualitativos convergem no reconhecimento de que o *Algoriddle* favorece a interpretação do comportamento interno dos algoritmos através da visualização passo-a-passo, da manipulação de parâmetros e do *feedback* imediato. Tal evidencia a resposta à primeira questão de investigação, relativa à perceção da usabilidade, clareza da interface e utilidade pedagógica do artefacto.

O tempo médio de 18 minutos e 34 segundos para completar o guião de tarefas e o questionário sugere que a aplicação é acessível e rápida de explorar por diferentes perfis de utilizadores, o que reforça a adequação da sua navegabilidade e organização visual. Ainda assim, este indicador deve ser interpretado com cautela: se, por um lado, evidencia ausência de dificuldades significativas, por outro poderá também indicar que algumas funcionalidades mais avançadas podem não ter sido exploradas de forma tão aprofundada quanto desejável durante a avaliação.

Do ponto de vista metodológico, o modelo DSR-E não só orientou a identificação do problema, a definição dos requisitos, o desenvolvimento do artefacto e a sua avaliação, como

também assegurou uma ligação sistemática entre as decisões de *design* e as necessidades educativas identificadas ao longo do processo. A estrutura cíclica do modelo permitiu analisar continuamente as evidências recolhidas, fundamentando ajustes e melhorias que contribuíram para a maturação do protótipo. Esta dinâmica comprova que o DSR-E desempenhou um papel determinante na coerência e evolução do *Algoriddle*, permitindo afirmar com segurança que a segunda questão de investigação foi respondida de forma positiva e com suporte empírico.

Em síntese, os resultados sugerem que o *Algoriddle* apresenta um grande potencial para o apoio à aprendizagem de algoritmos, embora não permita afirmar de forma categórica efeitos diretos sobre a aprendizagem autónoma ou os resultados académicos dos estudantes.

6.2 Limitações

Apesar dos resultados positivos, importa reconhecer algumas limitações inerentes ao estudo. A primeira prende-se com a dimensão da amostra, que embora relevante para a natureza exploratória da investigação, é reduzida e composta maioritariamente por estudantes e profissionais da área, o que pode influenciar a perceção da usabilidade e não permite generalizações amplas. Acresce que a investigação não incluiu docentes, limitando a análise da integração pedagógica em contexto curricular.

A ausência de recolha de outros dados sociodemográficos, impossibilita a análise de variáveis adicionais que poderiam influenciar a experiência de utilização, como o ano de curso ou o contacto prévio com ferramentas de visualização de algoritmos. Não foram realizadas medições de impacto direto no desempenho académico, dado que tal exigiria um estudo longitudinal em ambiente de sala de aula.

O conjunto de algoritmos disponível, embora adequado para esta fase, permanece limitado, restringindo a amplitude da análise e não cobrindo a totalidade dos conteúdos usualmente abordados em unidades curriculares de algoritmia. Por fim, a avaliação centrou-se predominantemente em perceções subjetivas, pelo que estudos futuros poderão beneficiar da integração de métricas mais objetivas, como tempos de execução de tarefas específicas, análise de erros ou monitorização de padrões de interação.

Estas limitações não invalidam os resultados, mas indicam oportunidades claras para investigação futura e aprofundamento do tema.

6.3 Trabalhos futuros

A investigação abre espaço a diversas linhas de continuidade que poderão ampliar o impacto do *Algoriddle* enquanto artefacto educativo. Embora nesta fase o foco tenha estado orientado sobretudo para o apoio aos estudantes, será importante envolver docentes em avaliações futuras, permitindo compreender de que forma a aplicação poderá ser integrada de forma eficaz nas práticas pedagógicas reais e qual o seu contributo no processo de ensino-aprendizagem. Os comentários recolhidos durante a avaliação também ofereceram contributos valiosos para a evolução da ferramenta, destacando a necessidade de otimizar o editor de código, melhorar os tempos de resposta, tornar o *feedback* visual mais imediato e aperfeiçoar aspetos da *interface* de modo a garantir uma interação mais fluida. Foram igualmente sugeridas funcionalidades com potencial para reforçar a motivação e a autonomia dos utilizadores, como desafios graduais, quizzes, exemplos práticos adicionais e registo de progresso, que podem incentivar uma exploração mais aprofundada e contínua da aplicação.

Num plano de expansão mais alargado, uma das intenções futuras consiste em aumentar a diversidade de algoritmos e conceitos disponíveis, incluindo outros conteúdos habitualmente abordados no ensino superior em Portugal, bem como temas que reflitam tendências emergentes, como algoritmos aplicados à Inteligência Artificial, contribuindo para a atualização e relevância do sistema perante os desafios atuais da computação. Reconhece-se igualmente a necessidade de reforçar a acessibilidade, incorporando funcionalidades direcionadas para utilizadores invisuais ou com limitações sensoriais, tais como suporte a leitores de ecrã e descrições alternativas, tornando o *Algoriddle* mais inclusivo.

Adicionalmente, a integração de métricas objetivas de aprendizagem e sistemas de personalização adaptativa poderá permitir ajustar o nível de detalhe das explicações ao perfil de cada utilizador, oferecendo uma experiência mais individualizada e fornecendo aos docentes informações relevantes para a identificação de dificuldades específicas. A interoperabilidade com plataformas de gestão de aprendizagem (LMS), como o *Moodle*, também surge como um passo natural para garantir maior adoção institucional e assegurar a sustentabilidade do projeto a longo prazo. De forma global, estas linhas de evolução poderão consolidar o *Algoriddle* como uma solução educativa mais completa, escalável e alinhada com as exigências contemporâneas do ensino da computação.

6.4 Considerações finais

O *Algoriddle* poderá representar um contributo relevante para o ensino e aprendizagem de algoritmos, ao proporcionar um ambiente visualmente apelativo, tecnicamente consistente e pedagogicamente fundamentado. A avaliação realizada sugere que o artefacto responde de forma adequada às necessidades identificadas e oferece uma experiência de utilização intuitiva e motivadora.

Apesar das limitações, os resultados indicam que o sistema possui potencial para integração em contextos formais e informais de aprendizagem, podendo ser utilizado tanto em unidades curriculares de algoritmia como recurso complementar, quanto de forma autónoma para prática individual. O trabalho desenvolvido reforça a importância das abordagens visuais e interativas no ensino de conteúdos abstratos, contribuindo para uma aprendizagem mais significativa e para a consolidação de bases sólidas de raciocínio computacional, sem permitir afirmar de forma categórica efeitos diretos sobre resultados académicos específicos.

Referências

A. Dresch, D. P. Lacerda, & J. A. V. Antunes. (2015). Design science research: Método de pesquisa para avanço da ciência e tecnologia. Porto Alegre: Bookman.

ACM/IEEE Joint Task Force on Computing Curricula. (2023). Computer science curricula 2023: Curriculum guidelines for undergraduate degree programs in computer science. ACM; IEEE Computer Society. https://ieeecs-media.computer.org/media/education/reports/CS2023.pdf (accessed Apr. 18, 2025)

Bangor, A., Kortum, P. T., & Miller, J. T. (2008). An empirical evaluation of the System Usability Scale. International Journal of Human–Computer Interaction, 24(6), 574–594.

Ben-Bassat Levy, R., Ben-Ari, M. & Uronen, P. A. (2003). *The Jeliot 2000 program animation system*. Computers & Education, 40(1), 1–15.

Brooke, J. (1996). SUS: A 'quick and dirty' usability scale. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, & I. L. McClelland (Eds.), Usability evaluation in industry (pp. 189–194). London, UK: Taylor & Francis.

Byrne, M. D., Catrambone, R., & Stasko, J. T. (1999). Evaluating animations as student aids in learning computer algorithms. Computers & Education, 33(4), 253–278.

Express. (n.d.). Node.js web application framework. https://expressjs.com/ (accessed Jul. 19, 2025)

Faculdade de Ciências da Universidade de Lisboa. https://fenix.ciencias.ulisboa.pt/degrees/engenharia-informatica-564500436615278/disciplina-curricular/564680825258135 (accessed Jan. 5, 2025)

Grover, S., Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42(1), 38–43.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. MIS Quarterly, 28(1), 75–105

Hohenwarter, M., & Gerber, M. (2009). Dynamic mathematics with GeoGebra. International Journal for Technology in Mathematics Education, 16(3), 89–100.

Hundhausen, C. D., Douglas, S. A., & Stasko, J. T. (2002). A meta-study of algorithm visualization effectiveness. Journal of Visual Languages & Computing, 13(3), 259–290.

i18next. (n.d.). Introduction | i18next documentation. https://www.i18next.com/ (accessed Jul. 19, 2025)

IEEE. (2011). ISO/IEC/IEEE 29148:2011 — Systems and software engineering — Life cycle processes — Requirements engineering. IEEE Standards Association.

Instituto Politécnico de Beja. (n.d.). https://www.ipbeja.pt/cursos/estig-ei/Descritores2015/9119118DPT.pdf (accessed Jan. 5, 2025)

Instituto Politécnico de Bragança. (n.d.). https://portalold.ipb.pt/index.php/pt/guiaects/cursos/licenciaturas/curso?cod_escola=3043&cod_curso=9119 (accessed Jan. 5, 2025)

Instituto Politécnico do Cávado e Ave. (n.d.). https://est.ipca.pt/curso/engenharia-informatica-medica/ (accessed Jan. 5, 2025)

Instituto Politécnico de Castelo Branco. (n.d.). https://www.ipcb.pt/estudar/cursos/licenciatura/licenciatura-em-engenharia-informatica/ (accessed Jan. 5, 2025)

Instituto Politécnico de Coimbra. (n.d.). https://www.ipc.pt/oferta-formativa/instituto-superior-de-engenharia-de-coimbra/licenciatura/licenciatura-em-engenharia-informatica/?d=254889&r=22704 (accessed Jan. 5, 2025)

Instituto Politécnico da Guarda. (n.d.). https://www.ipg.pt/guia_aluno/curso.aspx?id=4&curso=Engenharia%20Inform%C3%A1tica (accessed Jan. 5, 2025)

Instituto Politécnico de Leiria. (n.d.). https://www.ipleiria.pt/curso/licenciatura-em-engenharia-informatica (accessed Jan. 5, 2025)

Instituto Politécnico de Portalegre. (n.d.). https://www.ipportalegre.pt/en/oferta-formativa/informatica (accessed Jan. 5, 2025)

Instituto Politécnico do Porto. (n.d.). https://www.isep.ipp.pt/Course/Course/26 (accessed Jan. 5, 2025)

Instituto Politécnico de Santarém. (n.d.). https://academicos.ipsantarem.pt/disciplinas_geral.formview?p_cad_codigo=LIB10142&pv_periodo_pe=1S&p_ano_lectivo=2024 (accessed Jan. 5, 2025)

Instituto Politécnico de Setúbal. (n.d.). https://portal.ips.pt/ests/pt/ucurr_geral.ficha_uc_view?pv_ocorrencia_id=559643 (accessed Jan. 5, 2025)

Instituto Politécnico de Tomar. (n.d.). https://portal2.ipt.pt/pt/Cursos/licenciaturas/l_-_ei/911912/ (accessed Jan. 5, 2025)

Instituto Politécnico de Viana do Castelo. (n.d.). https://www.ipvc.pt/cursos/engenharia-informatica/?tab=planoestudos (accessed Jan. 5, 2025)

Instituto Politécnico de Viseu. (n.d.).

https://www1.estgv.ipv.pt/estgv/?fichaunidadecurricular&iduc=354&idc=9119&idp=2&estudar (accessed Jan. 5, 2025)

International Organization for Standardization. (2019). *ISO 9241-210:2019 – Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems*.

ISCTE-IUL. (n.d.). https://www.iscte-iul.pt/curso/1491/licenciatura-engenharia-informatica/planoestudos (accessed Jan. 5, 2025)

Kasanen, J., Lukka, K., & Siitonen, A. (1991). The constructive approach in management accounting research. Journal of Management Accounting Research, 3, 241–268.

Light City Medium. (n.d.). Data structure mastery unleashed: 8 dynamic online visualization platforms. https://light-city.medium.com/data-structure-mastery-unleashed-8-dynamic-online-visualization-platforms-fdc62bcc1225 (accessed Jan. 19, 2025)

Mayer, R. E. (2021). Multimedia learning (3rd ed.). Cambridge University Press.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., & Velázquez-Iturbide, J. Á. (2002). *Exploring the role of visualization and engagement in computer science education*. *SIGCSE Bulletin*, 35(2), 131–152.

Node.js. (n.d.). Don't block the event loop (or the worker pool). https://nodejs.org/id/learn/asynchronous-work/dont-block-the-event-loop (accessed Jul. 19, 2025)

Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. Basic Books.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. Journal of Management Information Systems, 24(3), 45–77.

Pimentel, M., Filippo, D., & dos Santos, T. M. (2020). Design science research: Pesquisa científica atrelada ao design de artefatos. RE@D - Revista de Educação a Distância e Elearning, 3(1).

Piston API. (n.d.). Piston: A high performance general purpose code execution engine. https://github.com/engineer-man/piston (accessed Jul. 30, 2025)

React. (n.d.). React – A JavaScript library for building user interface. https://react.dev/ (accessed Jul. 19, 2025)

Shaffer, C. A., Cooper, M. L., & Edwards, S. H. (2007). Algorithm visualization: A report on the state of the field. ACM SIGCSE Bulletin, 39(1), 150–154.

Simon, H. A. (1981). The sciences of the artificial (2nd ed.). MIT Press.

Sweller, J. (2011). Cognitive load theory (2nd ed.). Routledge.

Teodoro, V. D. (2002). Modellus: Learning physics with mathematical modelling.

TypeScript. (n.d.). TypeScript: JavaScript with syntax for types. https://www.typescriptlang.org/

UNESCO. (2017). A Guide for Ensuring Inclusion and Equity in Education. https://unesdoc.unesco.org/ark:/48223/pf0000248254 (accessed Sep. 30, 2025)

Universidade dos Açores – Faculdade de Ciências e Tecnologia. (n.d.). https://fct.uac.pt/disciplinas/85485-algoritmos-e-estruturas-de-dados/ (accessed Jan. 5, 2025)

Universidade do Algarve. (n.d.). https://www.ualg.pt/curso/1478/plano (accessed Jan. 5, 2025)

Universidade de Aveiro. (n.d.). https://www.ua.pt/pt/uc/12281 (accessed Jan. 5, 2025)

Universidade da Beira Interior. (n.d.). https://www.ubi.pt/Disciplina/14335/2024 (accessed Jan. 5, 2025)

Universidade de Coimbra. (n.d.). https://apps.uc.pt/courses/PT/unit/9854/25081/2025-2026?common_core=true&type=ram&id=362 (accessed Jan. 5, 2025)

Universidade de Évora. (n.d.). https://www.uevora.pt/estudar/cursos/licenciaturas?cod=9119&v=plano-estudos&uc=INF13188L (accessed Jan. 5, 2025)

Universidade de Lisboa – Técnico. (n.d.). https://fenix.tecnico.ulisboa.pt/cursos/leic-a/disciplina-curricular/845953938489544 (accessed Jan. 5, 2025)

Universidade da Madeira. (n.d.). https://www.uma.pt/ensino/1o-ciclo/licenciatura-em-engenharia-informatica/91958/?contentid=91958 (accessed Jan. 5, 2025)

Universidade do Minho. (n.d.). https://www.uminho.pt/pt/ensino/oferta-educativa/cursos-conferentes-a-grau/_layouts/15/uminho.portalum.ui/pages/catalogocursodetail.aspx?itemid=4079&catid=12 (accessed Jan. 5, 2025)

Universidade NOVA de Lisboa. (n.d.). https://guia.unl.pt/pt/2023/fct/program/1053/course/11154#subject (accessed Jan. 5, 2025)

Universidade do Porto – FEUP. (n.d.). https://sigarra.up.pt/feup/pt/ucurr_geral.ficha_uc_view?pv_ocorrencia_id=541876 (accessed Jan. 5, 2025)

Vite. (n.d.). Vite | Next Generation Front-End Tooling. https://vite.dev/ (accessed Jul. 19, 2025)

Wieman, C., Perkins, K., & Gilbert, S. (2012). Transforming physics education. Physics Today, 65(7), 36–41.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.

World Wide Web Consortium (W3C). (2018). Web Content Accessibility Guidelines (WCAG) 2.1. https://www.w3.org/TR/WCAG21/ (accessed Aug. 14, 2025)

Apêndices

Apêndice 1 – Manual do utilizador

1. Introdução

Este manual tem como objetivo fornecer orientações detalhadas sobre a utilização da aplicação de visualização de algoritmos, desenvolvida no âmbito de uma dissertação de mestrado. A aplicação foi concebida para apoiar o processo de ensino e aprendizagem, permitindo explorar algoritmos de forma interativa, visualizar a sua execução passo a passo, personalizar dados de entrada, comparar métodos e compreender conceitos teóricos associados.

O público-alvo inclui principalmente estudantes e docentes da área de informática e computação, mas a aplicação pode igualmente ser utilizada por qualquer pessoa interessada em aprofundar conhecimentos sobre algoritmos.

2. Requisitos do sistema

Navegador suportado: *Chrome*, *Firefox* e *Edge*.

Sistema operativo: *Windows*, *macOS* ou *Linux*.

Requisitos adicionais: ligação à Internet para atualizações e compatibilidade com ficheiros *CSV*/*JSON*.

3. Acesso à aplicação

1. Abra o navegador de Internet suportado.

2. Aceda ao endereço disponibilizado da aplicação (https://algoriddle.netlify.app/).

4. Estrutura da *interface*

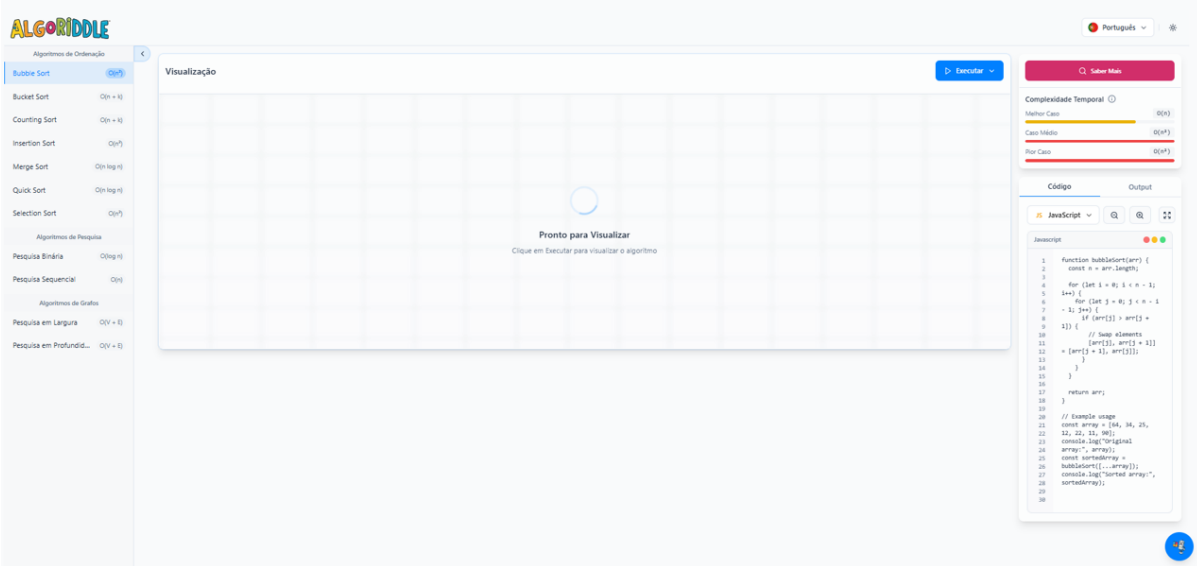


Figura 1 - Estrutura da *interface*

A interface da aplicação está organizada em **cinco áreas principais**, de forma a permitir uma utilização clara e intuitiva:

1. **Menu principal:** permite selecionar a categoria de algoritmos a explorar (ordenação, pesquisa, grafos).
2. **Área de visualização e controlo:** combina a visualização em tempo real da execução dos algoritmos, com suporte para zoom e expansão, com o painel de controlo. Esta área permite iniciar, pausar, retroceder e avançar na execução, além de ajustar a visualização conforme necessário.
3. **Painel informativo:** apresenta conteúdos teóricos e complementares, incluindo explicações textuais, análise de complexidade temporal e espacial, uso de memória, factos históricos e curiosidades.
4. **Secção de código:** fornece templates prontos em C, Java, Python e JavaScript, permitindo ao utilizador editar e executar código diretamente na aplicação, de forma integrada com a visualização dos algoritmos.
5. **Secção de apoio inteligente:** inclui o ***chatbot* integrado (*ChatGPT*)**, que responde a dúvidas em linguagem natural, e a **secção de *FAQs***, que apresenta perguntas e respostas frequentes sobre os algoritmos.

5. Funcionalidades principais

5.1 Algoritmos disponíveis

- **Ordenação (*Sorting*):** *Bubble Sort, Bucket Sort, Counting Sort, Insertion Sort, Merge Sort, Quick Sort* e *Selection Sort.*
- **Pesquisa (*Searching*):** Pesquisa binária e pesquisa sequencial.
- **Grafos (*Graph*):** Pesquisa em profundidade (DFS) e pesquisa em largura (BFS).

5.2 Visualização Interativa

A **área de visualização e controlo** oferece ao utilizador a capacidade de interagir com a execução dos algoritmos de forma clara e intuitiva, permitindo o controlo detalhado da execução e a visualização em tempo real. A secção está organizada conforme os tipos de algoritmos e opções de execução.

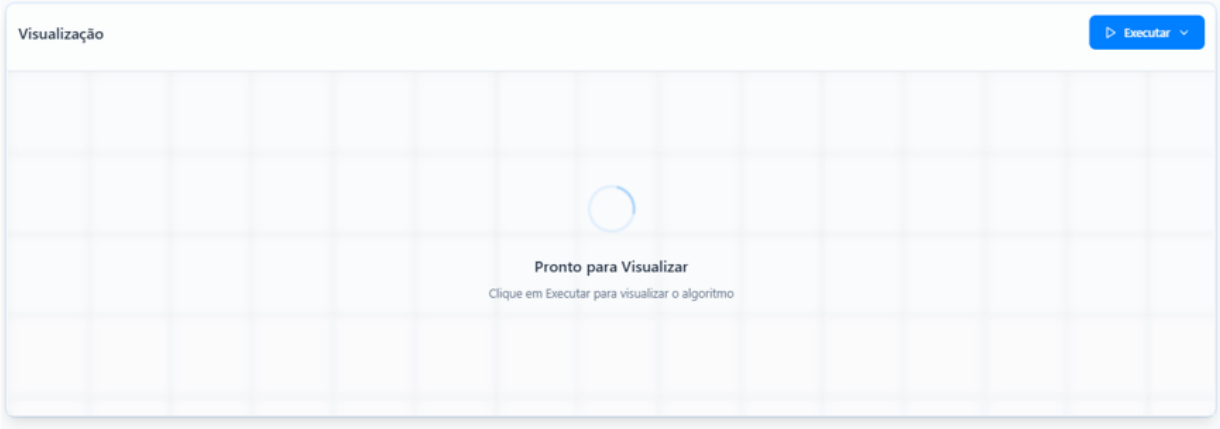


Figura 2 - Visualização interativa

5.2.1 Algoritmos de *ordenação* e *pesquisa*

Para os algoritmos de **ordenação** (sorting) e **pesquisa** (searching), o utilizador pode escolher entre diferentes formas de executar os algoritmos, com várias opções para configurar os dados de entrada e a execução:

- **Execução com Dados Aleatórios**: A aplicação gera automaticamente uma lista de números inteiros aleatórios, que são usados para testar os algoritmos. Esta opção permite testar os algoritmos com dados gerados aleatoriamente.
- **Execução com Dados Personalizados**: O utilizador pode:
    - **Criar listas manualmente**: O utilizador pode criar listas de números ou strings à mão, ajustando os dados conforme necessário.

- **Importar ficheiros CSV ou JSON**: Para cenários mais complexos, o utilizador pode importar dados de ficheiros CSV ou JSON, selecionando colunas ou campos específicos para utilização.

Além disso, em ambas as opções de dados (aleatórios ou personalizados), o utilizador pode **executar os algoritmos diretamente através do código**. A execução pode ser controlada e visualizada em tempo real, permitindo que o utilizador interaja com a animação passo a passo enquanto observa os resultados.

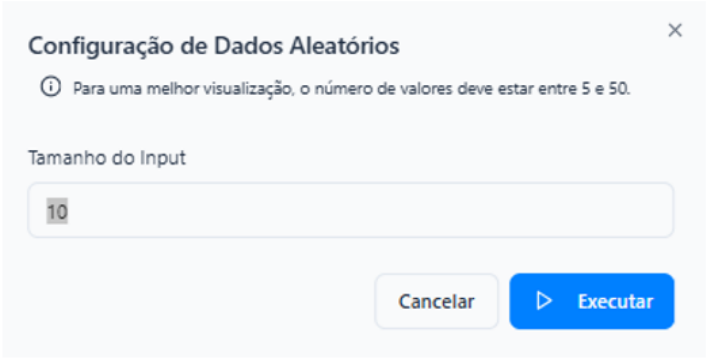


Figura 3 - Execução com dados aleatórios

Figura 4 - Execução com dados personalizados

5.2.3 Algoritmos de grafos

Tal como nos algoritmos de ordenação e pesquisa, os algoritmos de grafos também permitem a execução através do código. No entanto, ao contrário de sorting e searching, não é possível gerar dados aleatórios ou dados personalizados para os grafos. Em vez disso, o utilizador pode **construir o grafo visualmente**, selecionando nós e conectando-os conforme necessário.

Além disso, o utilizador pode **escolher o nó alvo para realizar a pesquisa**, definindo qual nó deseja investigar ou em que nó deseja iniciar a execução do algoritmo. A execução do algoritmo pode ser controlada diretamente a partir do código, com a visualização em tempo real da execução do grafo.

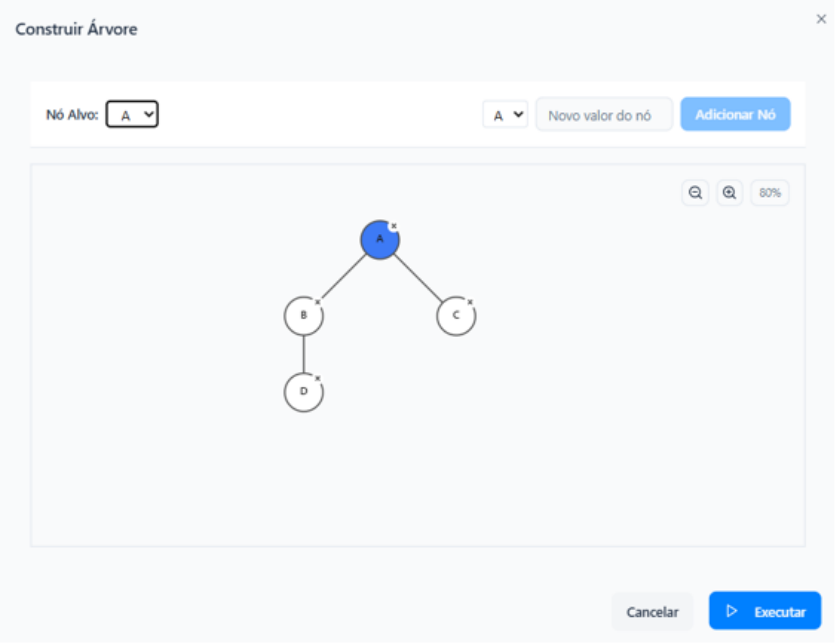


Figura 5 - Construção da árvore

5.2.4 Controlos de execução

O utilizador tem as seguintes opções para gerir a execução dos algoritmos:

- **Botões de reproduzir/pausar**: inicia ou interrompe a execução do algoritmo.
- **Botões de retroceder e avançar**: Permitem retroceder ou avançar etapas da execução, proporcionando maior controlo sobre o processo.
- **Zoom e expandir**: Ajustam a visualização do algoritmo, permitindo focar em detalhes específicos.
- **Botão de reiniciar**: Reinicia a execução do algoritmo, permitindo começar novamente do início.
- ***Slider* de velocidade**: Ajusta a velocidade da animação da execução, de acordo com a preferência do utilizador.

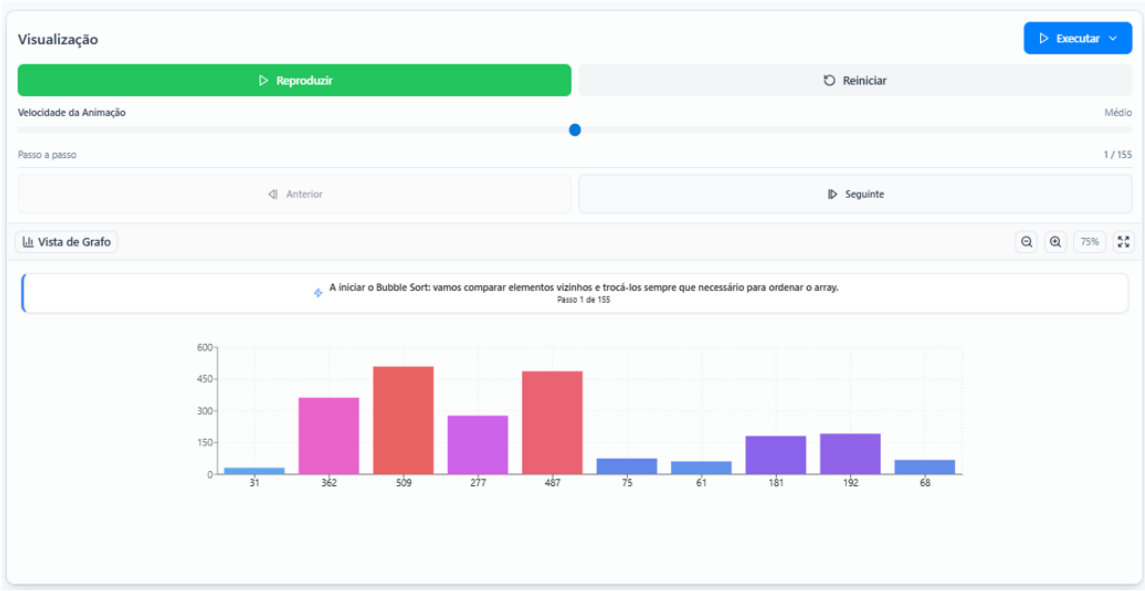


Figura 6 - Controlos de execução

5.2.5 Descrição didática

Cada algoritmo na aplicação vem acompanhado de uma **descrição didática e informativa**, que guia o utilizador através de cada etapa da execução. Essa descrição tem como objetivo garantir que os utilizadores compreendam claramente o funcionamento de cada algoritmo enquanto o visualizam em ação. Durante a execução, à medida que cada passo é realizado, uma explicação detalhada é apresentada, ajudando o utilizador a entender não apenas o "o que", mas também o "porquê" de cada ação.

As descrições incluem detalhes sobre o processo de execução, o comportamento dos dados em cada etapa, e as implicações das escolhas feitas durante a execução do algoritmo. Isso permite que o utilizador desenvolva uma compreensão mais profunda da lógica por trás de cada algoritmo, desde os métodos de ordenação e pesquisa até a manipulação de grafos, tornando a aprendizagem mais interativa e envolvente.

5.3 Painel informativo

No canto superior direito da aplicação encontra-se o **painel informativo**, um recurso valioso que oferece uma série de conteúdos teóricos e complementares para ajudar o utilizador a compreender melhor os algoritmos em execução. Este painel exibe informações detalhadas sobre cada algoritmo, incluindo:

Complexidade temporal: o painel mostra a complexidade temporal de cada algoritmo em três cenários principais — **melhor caso**, **pior caso** e **caso médio**:

- **Melhor caso**: é quando o algoritmo executa de forma mais eficiente, ou seja, o tempo de execução é o mais rápido possível. Geralmente acontece quando os dados estão já organizados de forma ideal.
- **Pior caso**: representa o cenário em que o algoritmo leva mais tempo para executar, ou seja, o pior desempenho possível. Isso ocorre quando os dados estão dispostos de forma que o algoritmo precise fazer mais operações.

- **Caso médio**: reflete o desempenho do algoritmo em situações comuns, com dados aleatórios ou típicos. Ele indica o tempo de execução esperado na maioria dos casos.

Figura 7 - Painel informativo

Além disso, o painel inclui uma opção interativa que oferece **informações adicionais**, permitindo ao utilizador aprofundar-se nos seguintes aspetos:

A primeira ***tab*** oferece uma **explicação detalhada sobre o algoritmo**, dividida nos seguintes pontos:

- **Passos do algoritmo:** fornece uma explicação passo a passo dos processos executados pelo algoritmo, detalhando a sequência de operações e decisões tomadas durante a sua execução.
- **Breve descrição do algoritmo:** apresenta uma descrição concisa sobre o funcionamento geral do algoritmo, incluindo o objetivo que visa atingir, as técnicas utilizadas e a forma como resolve o problema em questão.
- **Contexto histórico:** Inclui informações sobre a origem e evolução do algoritmo ao longo do tempo, ajudando o utilizador a compreender o desenvolvimento histórico das técnicas envolvidas.
- **Factos interessantes:** Apresenta curiosidades sobre o algoritmo, como variações populares, exemplos de uso e informações adicionais que tornam o estudo mais enriquecedor.

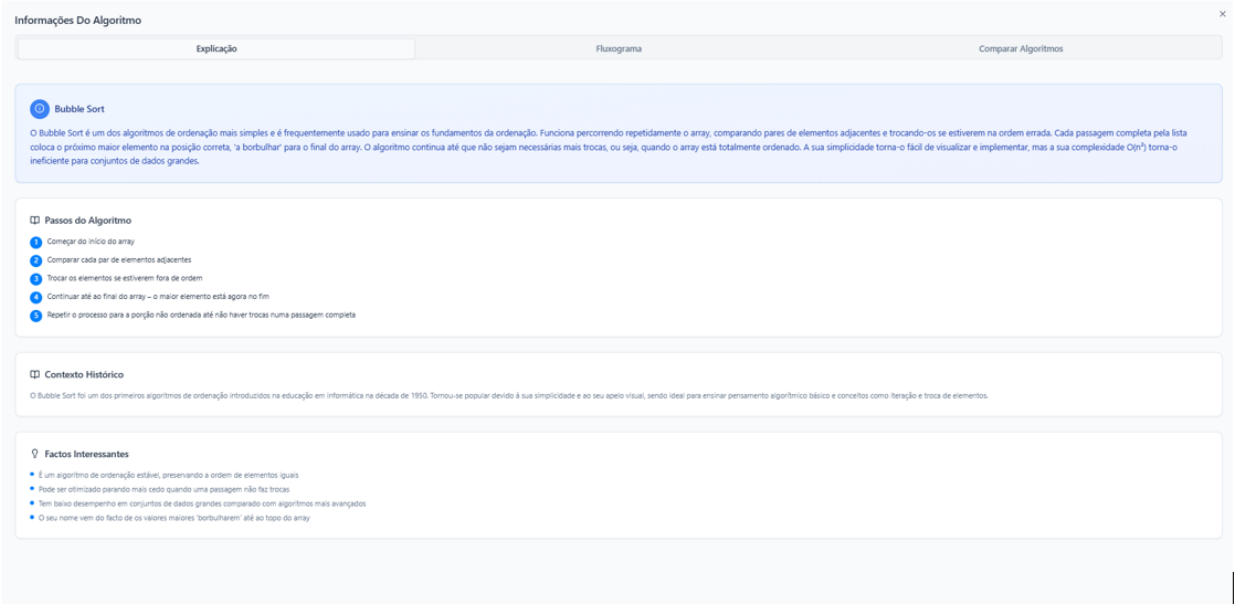


Figura 8 – Explicação detalhada do algoritmo

A aplicação oferece, para cada algoritmo, uma **representação gráfica em fluxograma** (*flowchart*), permitindo ao utilizador visualizar de forma clara e intuitiva o fluxo de execução passo a passo. Na **segunda *tab***, o fluxograma exibe a sequência lógica das operações de maneira simples e acessível, ajudando o utilizador a compreender como os dados são processados e como as decisões são tomadas em cada etapa do algoritmo. Esse recurso facilita a visualização da lógica de execução, tornando o entendimento do processo mais direto e intuitivo.

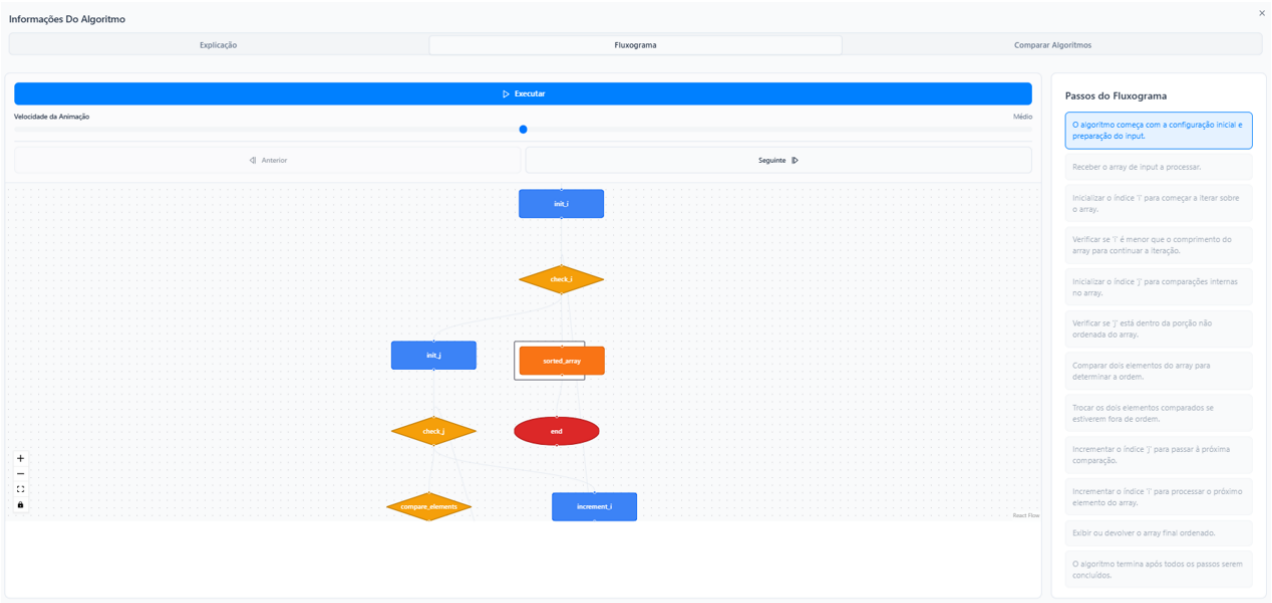


Figura 9 – Fluxograma

A terceira **tab** oferece uma funcionalidade que permite ao utilizador comparar diferentes algoritmos de forma eficaz. O **comparador de algoritmos** destaca as diferenças entre eles em diversos aspetos, incluindo:

- **Escalabilidade**: como o algoritmo se comporta à medida que o tamanho dos dados aumenta.
- **Complexidade temporal**: A eficiência do algoritmo em termos de tempo de execução, considerando os três cenários principais (melhor caso, pior caso e caso médio).
- **Uso de memória**: A quantidade de memória que o algoritmo requer durante a execução.
- **Outras observações**: inclui fatores adicionais que podem impactar o desempenho, como estabilidade, facilidade de implementação, entre outros.

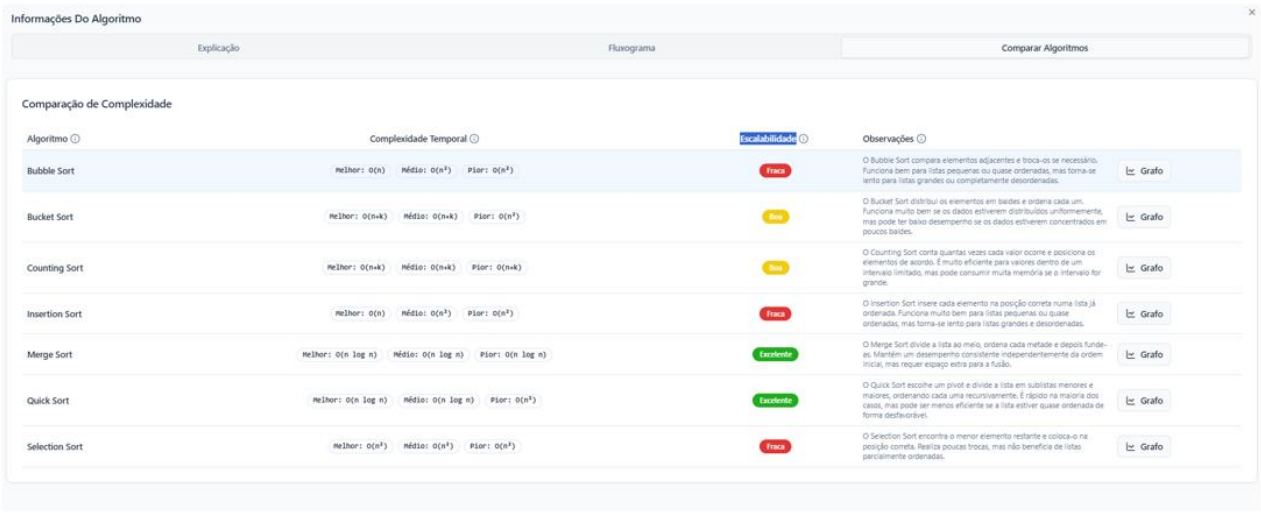


Figura 10 - Comparação de algoritmos

Além disso, na **tabela comparativa**, ao clicar na opção **grafo**, um ***popup*** é exibido, permitindo ao utilizador **selecionar diferentes algoritmos** e visualizar, de forma gráfica, como cada um deles se comporta. Isso proporciona uma comparação visual imediata, facilitando o entendimento do desempenho relativo de cada algoritmo.

Esta ferramenta ajuda o utilizador a avaliar a **eficiência** e a **adequação** de diferentes abordagens, permitindo escolher a mais eficaz para cada cenário específico.

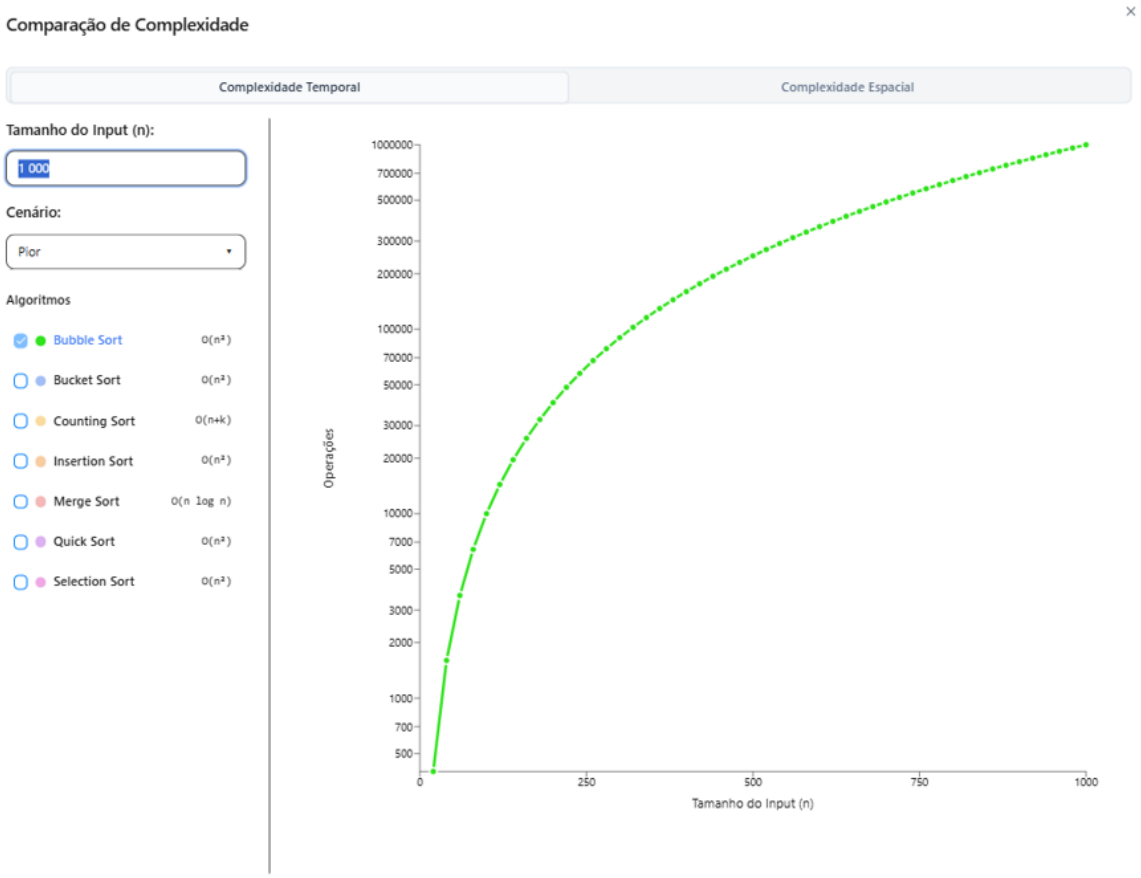


Figura 11 - Comparação visual

5.4 Editor de código

A aplicação inclui um **editor de código** integrado, que permite ao utilizador editar e personalizar os algoritmos diretamente na interface. Este editor oferece suporte para várias linguagens de programação, incluindo ***C***, ***Java***, ***Python*** e ***JavaScript***, proporcionando flexibilidade para programadores com diferentes preferências.

O editor é altamente interativo, permitindo que o utilizador:

- **Utilize templates prontos**: o editor oferece ***templates*** prontos de algoritmos, que o utilizador pode personalizar e testar rapidamente, sem precisar começar do zero.
- **Teste e depure o código**: o utilizador pode modificar o código, testá-lo e visualizar o impacto das mudanças na execução do algoritmo em tempo real.

Além disso, o editor está totalmente integrado com a área de **visualização e controlo**, permitindo ao utilizador ver, de imediato, como as suas alterações no código afetam a execução e o comportamento do algoritmo.

5.8 Multilinguismo

A aplicação encontra-se disponível em cinco idiomas: **português**, **inglês**, **espanhol**, **francês** e **alemão**. A alteração de idioma pode ser realizada no menu superior, garantindo acessibilidade a estudantes internacionais.

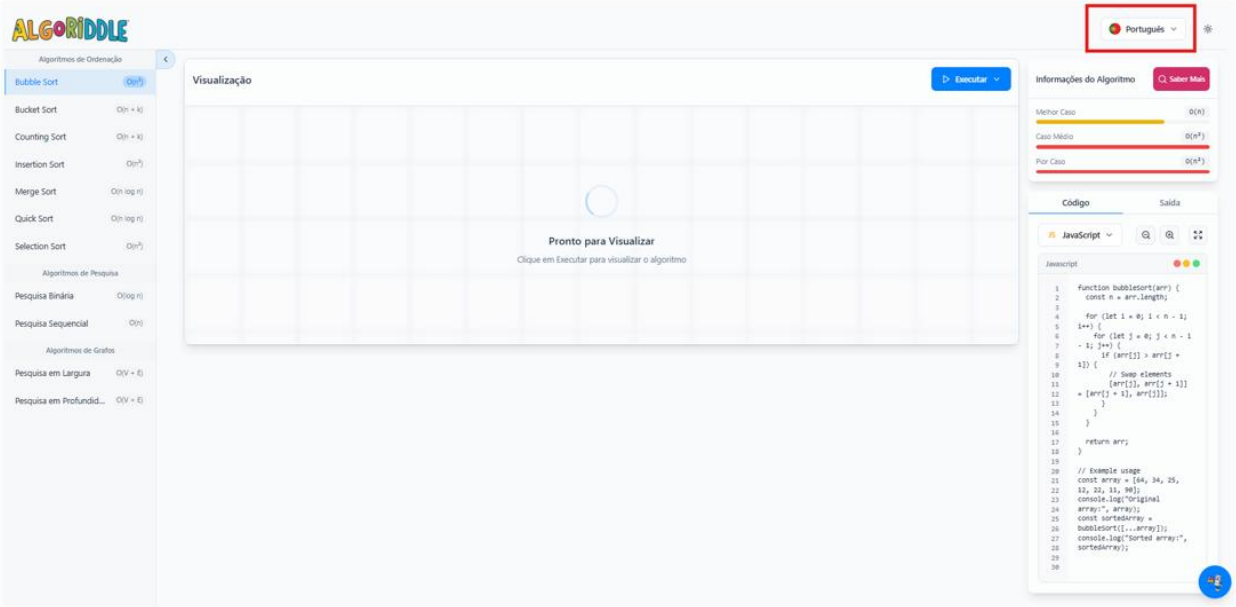


Figura 12 - Multilinguismo

5.9 *Chatbot* e *FAQs*

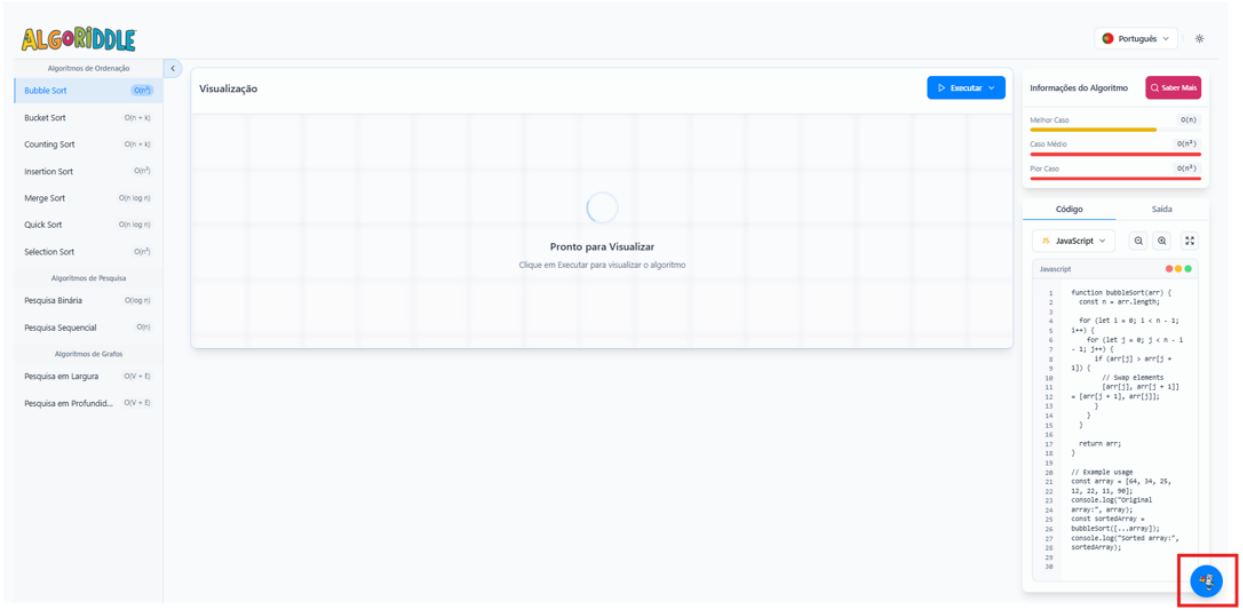


Figura 13 - Acesso ao *chatbot*

Figura 14 - *Chatbot*

A aplicação inclui uma **secção de apoio inteligente**, destinada a auxiliar o utilizador na compreensão dos algoritmos e na utilização da aplicação. Esta secção integra duas funcionalidades principais:

A aplicação inclui uma **secção de apoio inteligente**, destinada a auxiliar o utilizador na compreensão dos algoritmos e na utilização da aplicação. Esta secção integra duas funcionalidades principais:

Chatbot integrado (*ChatGPT*):

- Assistente virtual que permite ao utilizador colocar perguntas em linguagem natural sobre algoritmos.
- Permite esclarecer dúvidas em tempo real, tornando a aprendizagem mais interativa e personalizada.

***FAQs* (Perguntas frequentes):**

- Lista organizada de perguntas e respostas frequentes relacionadas com algoritmos e boas práticas de programação.
- Facilita a resolução de dúvidas comuns sem necessidade de recorrer a fontes externas.
- Atualizada dinamicamente com base nas interações dos utilizadores e nas questões mais recorrentes.

Apêndice 2 – Guião de tarefas e questionário de avaliação

Guião de Teste e Questionário de Avaliação da Aplicação

Este documento reúne o **guião de tarefas** e o **questionário de avaliação** da aplicação **ALGORIDDLE**, com o objetivo de recolher opiniões de estudantes sobre três áreas principais:

- **Usabilidade**: Refere-se à facilidade com que os utilizadores interagem com a aplicação, avaliando a sua eficiência, intuitividade e acessibilidade.
- **Pertinência pedagógica**: Avalia a adequação dos conteúdos e funcionalidades da aplicação em relação aos objetivos educativos, considerando o alinhamento com o currículo e a eficácia no processo de ensino-aprendizagem.
- **Impacto na aprendizagem**: Examina como a utilização da aplicação afeta o desempenho e o envolvimento dos estudantes, assim como o seu impacto no processo de aprendizagem de uma forma geral.

Escala de resposta:

1 = Discordo totalmente
2 = Discordo
3 = Neutro
4 = Concordo
5 = Concordo totalmente

🧩 **O Software ALGORIDDLE**

O **ALGORIDDLE** é uma aplicação educativa interativa concebida para apoiar o ensino de **algoritmos** de forma clara e envolvente. Através de **visualizações animadas**, **explicações detalhadas** e **exemplos práticos** em várias linguagens de programação, facilita a compreensão de temas como ordenação, pesquisa e grafos.

A aplicação:

- **Pretende tornar o estudo de algoritmos mais acessível e dinâmico**.
- **Pretende estimular o pensamento crítico, a autonomia e o raciocínio lógico**.
- **Pretende promover uma aprendizagem gradual e significativa** dos fundamentos da computação.

Secção A — Perfil do Respondente

1. Experiência em programação: ( ) Iniciante ( ) Intermédio ( ) Avançado

---

Secção B — Guião de Tarefas

Nesta secção, será solicitado que realize uma série de tarefas utilizando o *software* **ALGORIDDLE**. Através de cada tarefa, terá a oportunidade de explorar diferentes funcionalidades e fornecer o seu *feedback* sobre a clareza das visualizações, a interação com os algoritmos e a experiência geral.

Importante: Durante a execução das tarefas, preste atenção à fluidez da aplicação, à clareza das explicações e à eficácia das ferramentas interativas (como pausa, retroceder e avançar). Após a conclusão de cada tarefa, será solicitado que avalie a sua experiência com base numa escala de 1 a 5.

No final de cada tarefa, poderá deixar comentários adicionais para detalhar a sua experiência.

Tarefa 1 – Navegação e Estrutura da Interface

Objetivo: Avaliar a clareza e a organização da interface.
Instruções:

1. Aceda à aplicação e explore o menu principal.
2. Identifique as cinco áreas principais (menu, área de visualização, painel informativo, secção de código e apoio inteligente).
3. Verifique se a disposição dos elementos é intuitiva.

Avaliação da tarefa:
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Comentários sobre esta tarefa:
_____
_____
_____
_____
_____

Tarefa 2 – Execução de um Algoritmo de Ordenação

Objetivo: Avaliar a usabilidade dos controlos de execução e clareza da visualização.
Instruções:

1. Execute o ***Bubble Sort*** com dados aleatórios.
2. Use os botões de pausa, retroceder, avançar e o controlo de velocidade.
3. Observe se compreende o que está a acontecer em cada passo da animação.

Avaliação da tarefa:
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Comentários sobre esta tarefa:
_____
_____
_____
_____
_____

Tarefa 3 – Personalização de Dados (Manual e JSON)

Objetivo: Avaliar a facilidade de personalização de dados e importação de ficheiros.
Instruções:

Inserção manual de dados personalizados

1. Escreva um conjunto de valores separados por vírgulas (ex.: 42, 17, 88, 3, 29).
2. Aplique o ***Bucket Sort*** a essa lista.
3. Observe e avalie a clareza da visualização animada do processo de ordenação.

Importação de ficheiro JSON com objetos

1. Importe um ficheiro *JSON* contendo objetos com múltiplos campos (ex.: nome, idade, nota, etc.).
2. Aplique o ***Bucket Sort***, selecionando um campo específico para ordenação (ex.: "nota").
3. Observe a visualização do algoritmo com esses dados estruturados.

Avaliação da tarefa:
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Comentários sobre esta tarefa:

Tarefa 4 – Comparação de Algoritmos

Objetivo: Avaliar a utilidade e clareza do painel comparativo.
Instruções:

1. Aceda às informações detalhadas de um algoritmo (por exemplo, ***Merge Sort***) e clique em **"Saber Mais"**.
2. Abra a aba ou secção **"Comparar Algoritmos"**.
3. No painel comparativo:
   - Adicione o ***Quick Sort*** para comparação.
   - Observe os gráficos e tabelas apresentados, incluindo:
     - **Complexidade temporal** (melhor, médio e pior caso).
     - **Utilização de memória** (eficiência espacial).
4. Analise os dados apresentados no gráfico comparativo:
   - A informação é clara e bem organizada?
   - A comparação facilita a compreensão das diferenças entre os algoritmos?

Avaliação da tarefa:
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Comentários sobre esta tarefa:

Tarefa 5 – Execução de Algoritmos de Pesquisa

Objetivo: Avaliar a compreensão dos algoritmos de pesquisa e a clareza da explicação e visualização associadas.
Instruções:

1. Execute os algoritmos de **pesquisa sequencial** e **pesquisa binária** através de dados aleatórios.
2. Compare:
   - O **número de passos** realizados até encontrar o elemento.
   - O **tempo de execução**, se apresentado.
   - A **animação do processo** de pesquisa (clareza e utilidade da visualização).
3. Refira se conseguiu compreender:
   - A lógica de funcionamento de cada algoritmo.
   - As **principais diferenças** entre a pesquisa sequencial e binária.

Avaliação da tarefa:
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Comentários sobre esta tarefa:

Tarefa 6 – Construção e Exploração de Grafos

Objetivo: Avaliar a interação visual e compreensão dos algoritmos de grafos.
Instruções:

1. Construa um **grafo em forma de árvore** (sem ciclos), com pelo menos 6 a 8 nós.
    - Exemplo: uma estrutura hierárquica ou de decisão simples.
    - Selecione um **nó de partida (nó raiz ou alvo)**.
2. Aplique os algoritmos:
    - **DFS (Pesquisa em Profundidade)**
    - **BFS (Pesquisa em Largura)**
3. Registe a **ordem de visita dos nós** em cada caso.
4. Analise:
    - A **clareza da visualização** (cores, animações, sequência).
    - Se as **diferenças entre DFS e BFS** são compreensíveis.
    - A **intuitividade da explicação** e representação do processo.

Avaliação da tarefa:
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Comentários sobre esta tarefa:

Tarefa 7 – Editor de Código Multilingue

Objetivo: Avaliar a integração do editor com a visualização e a execução em diferentes linguagens.
Instruções:

1. Abra o template de ***Quick Sort*** em ***Python*** e execute.
2. Faça o mesmo em ***Java***.
3. Edite o código (por exemplo, alterando o tamanho da lista) e verifique se a visualização responde corretamente.

Avaliação da tarefa:
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Comentários sobre esta tarefa:

Tarefa 8 – Conteúdo Didático e Painel Informativo

Objetivo: Avaliar a pertinência pedagógica das explicações teóricas e fluxogramas.
Instruções:

1. Aceda às informações detalhadas de um algoritmo (por exemplo, ***Selection Sort***) e clique em **"Saber Mais"**.
2. Abra as abas ou secções **"Explicação"** e **"Fluxograma"** .
3. Avalie a clareza, rigor e relevância das informações.

Avaliação da tarefa:
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Comentários sobre esta tarefa:

_____
_____
_____
_____
_____

Tarefa 9 – Multilinguismo

Objetivo: Avaliar a qualidade e consistência da tradução.
Instruções:

- Altere o idioma para **inglês** e depois para **espanhol**.
- Observe menus, botões e descrições.
- Verifique se há incoerências linguísticas ou perda de significado técnico.

Avaliação da tarefa:
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Comentários sobre esta tarefa:

_____
_____
_____
_____
_____

Tarefa 10 – Apoio Inteligente e FAQs

Objetivo: Avaliar a eficácia dos mecanismos de apoio ao utilizador.
Instruções:

- No **chatbot**, coloque uma questão sobre um algoritmo (ex.: "Como funciona o Quick Sort?").
- Depois, procure a mesma resposta na **secção de FAQs**.
- Compare clareza, utilidade e rapidez de resposta.

Avaliação da tarefa:
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Comentários sobre esta tarefa:

Secção C — Usabilidade

Nesta secção, gostaríamos de conhecer a sua opinião sobre a **usabilidade** da aplicação ALGORIDDLE. A sua avaliação será focada na facilidade de navegação, na clareza dos controlos de execução e na eficácia das visualizações gráficas. O seu feedback ajudará a identificar pontos fortes e áreas de melhoria, com o objetivo de tornar a aplicação mais intuitiva e fácil de usar para todos os utilizadores.

	1 Discordo totalmente	2 Discordo	3 Neutro	4 Concordo	5 Concordo totalmente
A interface é intuitiva e de fácil utilização.	☐	☐	☐	☐	☐
Os controlos de execução (pausa, retroceder, avançar, passo a passo) são claros e úteis.	☐	☐	☐	☐	☐
A navegação entre algoritmos e funcionalidades é rápida e consistente.	☐	☐	☐	☐	☐
As visualizações gráficas (animações e legendas) facilitam a compreensão.	☐	☐	☐	☐	☐
A aplicação respondeu de forma fluida e sem atrasos.	☐	☐	☐	☐	☐

Secção D — Impacto na Aprendizagem/Pedagogia

Nesta secção, gostaríamos de avaliar o impacto da aplicação ALGORIDDLE no processo de aprendizagem e no seu valor pedagógico. O objetivo é perceber de que forma as funcionalidades da aplicação contribuem para a compreensão dos algoritmos, o desenvolvimento de competências analíticas e a motivação dos estudantes. A sua opinião será valiosa para entender como a aplicação pode melhorar a experiência de ensino e aprendizagem de algoritmos.

	1 Discordo totalmente	2 Discordo	3 Neutro	4 Concordo	5 Concordo totalmente
A visualização passo a passo contribuiu para melhor compreender os algoritmos.	☐	☐	☐	☐	☐
A possibilidade de usar dados personalizados tornou a experiência mais significativa.	☐	☐	☐	☐	☐
As explicações teóricas e complexidade foram úteis e rigorosas.	☐	☐	☐	☐	☐
A comparação entre algoritmos (melhor, médio, pior caso) ajudou a perceber diferenças de desempenho.	☐	☐	☐	☐	☐
As funcionalidades multilingues são úteis para aprendizagem e ensino em contextos diversos.	☐	☐	☐	☐	☐

Secção E — Funcionalidades de Apoio

Nesta secção, queremos avaliar as funcionalidades de apoio disponíveis na aplicação ALGORIDDLE, como o chatbot, as FAQs e os templates de código em diferentes linguagens. A sua opinião ajudará a entender como estas ferramentas adicionais contribuem para esclarecer dúvidas, apoiar a prática de programação e reforçar o interesse pelo estudo dos algoritmos. O seu feedback será essencial para melhorar a eficácia e utilidade destas funcionalidades.

	1 Discordo totalmente	2 Discordo	3 Neutro	4 Concordo	5 Concordo totalmente
O chatbot forneceu respostas claras e relevantes.	☐	☐	☐	☐	☐
As FAQs são eficazes para esclarecer dúvidas frequentes.	☐	☐	☐	☐	☐
Os templates de código em diferentes linguagens apoiam a prática de programação.	☐	☐	☐	☐	☐
Os conteúdos de apoio (factos históricos, flowcharts) reforçam o interesse e a aprendizagem.	☐	☐	☐	☐	☐

Secção F — Recomendação e Motivação

Nesta secção, queremos avaliar as funcionalidades de apoio disponíveis na aplicação ALGORIDDLE, como o chatbot, as FAQs e os templates de código em diferentes linguagens. A sua opinião ajudará a entender como estas ferramentas adicionais contribuem para esclarecer dúvidas, apoiar a prática de programação e reforçar o interesse pelo estudo dos algoritmos. O seu feedback será essencial para melhorar a eficácia e utilidade destas funcionalidades.

	1 Discordo totalmente	2 Discordo	3 Neutro	4 Concordo	5 Concordo totalmente
A aplicação aumentou a motivação para explorar algoritmos.	☐	☐	☐	☐	☐
Recomendaria esta aplicação a outros estudantes.	☐	☐	☐	☐	☐
Vejo potencial de integração da aplicação em contextos de ensino/aprendizagem.	☐	☐	☐	☐	☐

Secção G — Comentários Abertos

Nesta secção, convidamo-lo a partilhar qualquer comentário adicional que considere relevante sobre a aplicação ALGORIDDLE. Pode sugerir melhorias, destacar funcionalidades que considere particularmente úteis ou inovadoras, ou ainda apontar qualquer outra observação que ajude a melhorar a experiência de uso e o impacto pedagógico da aplicação. O seu feedback é fundamental para o desenvolvimento contínuo da plataforma.

1. Qual considera a funcionalidade mais útil da aplicação?

   _____
   _____

2. Indique um aspeto positivo da aplicação.

   _____
   _____

3. Apresente um aspeto que considera prioritário para melhorar.

   _____
   _____
   _____

4. Sugira funcionalidades adicionais que poderiam aumentar a utilidade pedagógica da aplicação.

   _____
   _____

5. Qual considera a funcionalidade mais inovadora ou diferenciadora?

   _____
   _____

6. Que melhorias ou ajustes considera essenciais para otimizar a aplicação?

   _____
   _____
   _____

7. Comentários gerais sobre a aplicação:

   _____
   _____

Apêndice 3 – Estatísticas do questionário de avaliação

Responses Overview

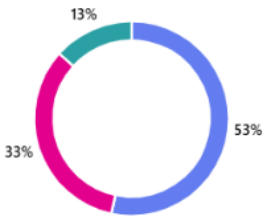
Active

Responses	Average Time	Duration
15	18:34	16 Days

1. Experiência em programação

More details

Iniciante	8
Intermédio	5
Avançado	2

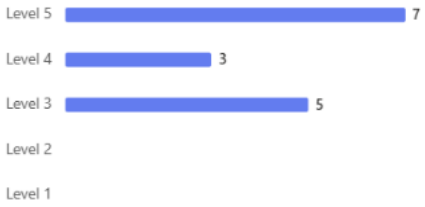


2. Avaliação da tarefa

More details

4.13

Average Rating



3. Comentários sobre a tarefa

More details

15

Responses

Latest Responses

"Estrutura moderna e limpa."

"Estrutura bem organizada, fácil de navegar."

"Layout simples, mas podia ter mais contraste."

...

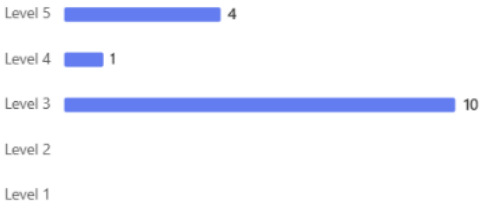
5 respondents (33%) answered Estrutura for this question.

4. Avaliação da tarefa

More details

3.60

Average Rating



5. Comentários sobre a tarefa

More details

15

Responses

Latest Responses

"Botões bem posicionados."

"Execução fluída."

"Botões funcionais, rápidos."

...

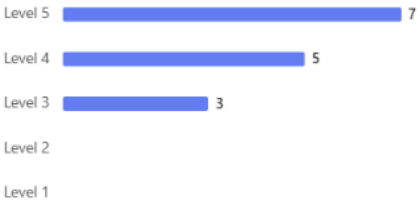
8 respondents (53%) answered Botões for this question.

6. Avaliação da tarefa

More details

4.27

Average Rating



7. Comentários sobre a tarefa

More details

12
Responses

Latest Responses
"JSON importado rápido."
...

9 respondents (75%) answered JSON for this question.

8. Avaliação da tarefa

More details

4.00
Average Rating

9. Comentários sobre a tarefa

More details

15
Responses

Latest Responses
"Comparador prático, mas precisa de cores."
"Comparador útil, faltam detalhes visuais."
"Comparador fácil de usar, faltam legendas."
...

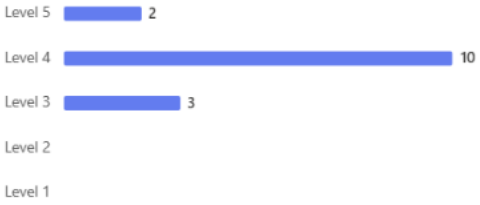
15 respondents (100%) answered Comparador for this question.

10. Avaliação da tarefa

More details

3.93

Average Rating



11. Comentários sobre a tarefa

More details

15

Responses

Latest Responses

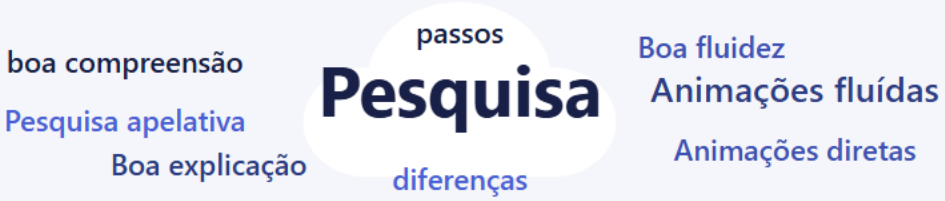
"Animações diretas."

"Pesquisa bem implementada."

"Animações fluídas."

...

6 respondents (40%) answered Pesquisa for this question.

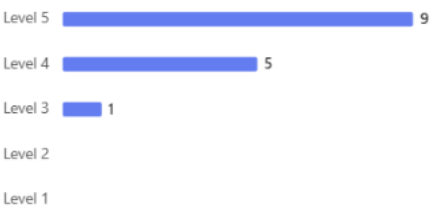


12. Avaliação da tarefa

More details

4.53

Average Rating



13. Comentários sobre a tarefa

More details

15
Responses

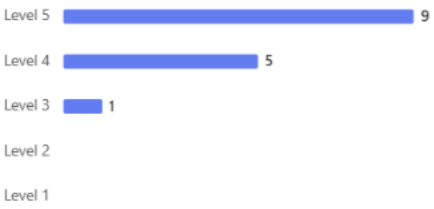
Latest Responses
"Boa interação."
"Boa interação geral."
"Boa interação geral."
...

5 respondents (33%) answered DFS for this question.

14. Avaliação da tarefa

More details

4.53
Average Rating



15. Comentários sobre a tarefa

More details

15
Responses

Latest Responses
"Execução fluída."
"Visualização intuitiva."
"Visualização clara."
...

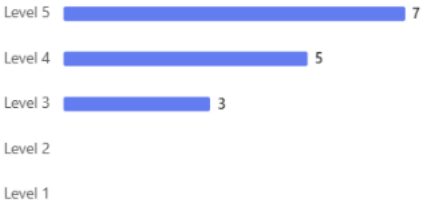
5 respondents (33%) answered Integração for this question.

16. Avaliação da tarefa

More details

4.27

Average Rating



17. Comentários sobre a tarefa

More details

15

Responses

Latest Responses

"Fluxogramas curtos, mas eficazes."

"Painel claro."

"Fluxogramas diretos."

...

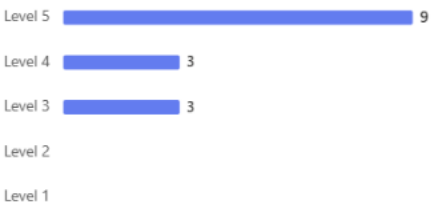
7 respondents (47%) answered Fluxogramas for this question.

18. Avaliação da tarefa

More details

4.40

Average Rating



19. Comentários sobre a tarefa

More details

15
Responses

Latest Responses
"Traduções boas."
"Traduções coerentes."
"Terminologia coerente."
...

9 respondents (60%) answered Traduções for this question.

20. Avaliação da tarefa

More details

3.53
Average Rating

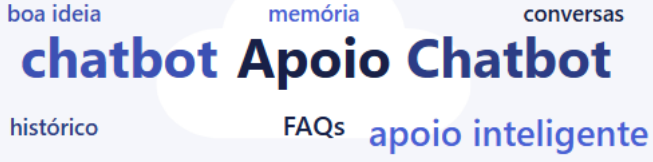
21. Comentários sobre a tarefa

More details

15
Responses

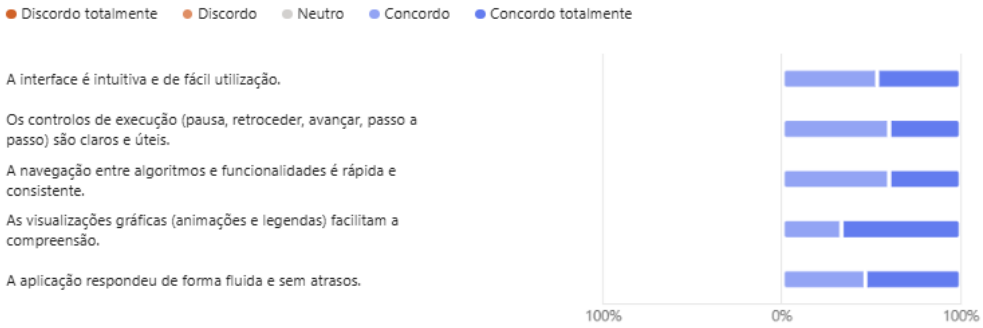
Latest Responses
"Chatbot funcional."
"Apoio inteligente prático."
"Chatbot rápido."
...

6 respondents (40%) answered Apoio for this question.



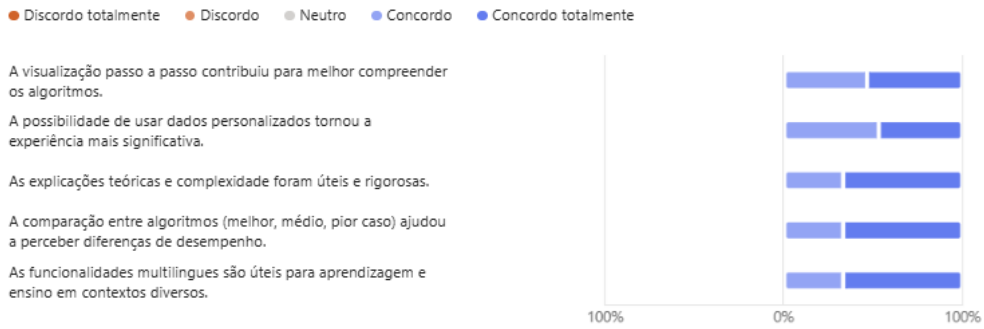
22. Usabilidade

More details



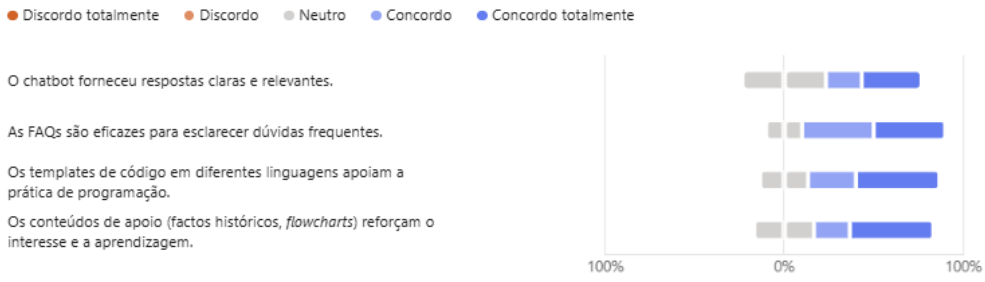
23. Impacto na Aprendizagem / Pedagogia

More details



24. Funcionalidades de Apoio

More details



25. Funcionalidades de Apoio

More details

26. Qual considera a funcionalidade mais útil da aplicação ?

More details

15
Responses

Latest Responses
"Importação de JSON é o ponto alto."
"As FAQs e apoio são úteis."
"Criar e explorar grafos foi divertido."
...

3 respondents (20%) answered Comparador for this question.

27. Indique um aspeto positivo da aplicação

More details

15
Responses

Latest Responses
"Interface simples e rápida."
"Descrições dos algoritmos muito boas."
"Animações suaves."
...

3 respondents (20%) answered Interface for this question.

28. Apresente um aspeto que considera prioritário para melhorar

More details

15
Responses

Latest Responses
"Legendas passam depressa."
"Faltam exemplos em mais linguagens."
"O menu podia ser mais segmentado."
...

2 respondents (13%) answered exemplos for this question.

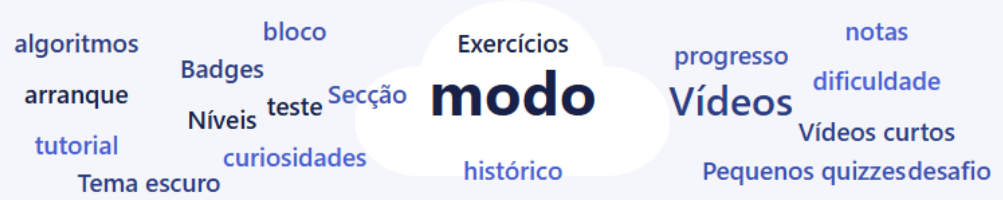
29. Sugira funcionalidades adicionais que poderiam aumentar a utilidade pedagógica da aplicação

More details

15
Responses

Latest Responses
"Secção de curiosidades seria fixe."
"Vídeos ajudavam a complementar."
"Níveis de dificuldade seriam engraçados."
...

3 respondents (20%) answered modo for this question.



30. Qual considera a funcionalidade mais inovadora ou diferenciadora ?

More details

15
Responses

Latest Responses
"Editor destaca-se pela utilidade."
"Visualização muito impressionante."
"Comparador de complexidade excelente."
...

4 respondents (27%) answered Comparador for this question.

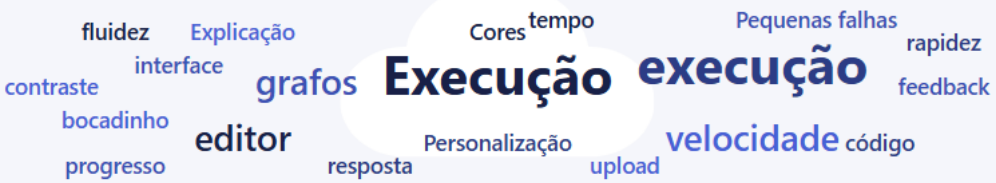
31. Que melhorias ou ajuste considera essenciais para otimizar a aplicação ?

More details

15
Responses

Latest Responses
"Personalização de interface seria top."
"Execução podia ser mais rápida."
"Ajustar velocidade seria bom."
...

3 respondents (20%) answered Execução for this question.




32. Comentários gerais sobre a aplicação
More details
15
Responses
Latest Responses
"App divertida e útil."
"Excelente experiência geral."
"App muito bem conseguida."
...
6 respondents (40%) answered App for this question.
Boa experiência
Gostei
qualidade
App
Excelente experiência
aprendizagem
dedicação
grande potencial
Ferramenta
geral
boa aplicação
trabalho
algoritmos

Apêndice 4 – Questionário SUS

Escala de resposta:

1 = Discordo totalmente
2 = Discordo
3 = Neutro
4 = Concordo
5 = Concordo totalmente

1 - Acho que gostaria de usar este sistema com frequência.

☐1 ☐2 ☐3 ☐4 ☐5

2 - Achei o sistema desnecessariamente complexo.

☐1 ☐2 ☐3 ☐4 ☐5

3 - Achei o sistema fácil de usar.

☐1 ☐2 ☐3 ☐4 ☐5

4 - Acho que precisaria de ajuda de alguém experiente para usar este sistema.

☐1 ☐2 ☐3 ☐4 ☐5

5 - Achei que as várias funções do sistema estavam bem integradas.

☐1 ☐2 ☐3 ☐4 ☐5

6 - Achei que havia demasiada inconsistência no sistema.

☐1 ☐2 ☐3 ☐4 ☐5

7 - Imagino que a maioria das pessoas aprenderia a usar este sistema muito rapidamente.

☐1 ☐2 ☐3 ☐4 ☐5

8 - Achei o sistema muito pesado de usar.

☐1 ☐2 ☐3 ☐4 ☐5

9 - Senti-me bastante confiante a usar o sistema.

☐1 ☐2 ☐3 ☐4 ☐5

10 - Necessitava de aprender muitas coisas antes de poder usar o sistema.

☐1 ☐2 ☐3 ☐4 ☐5